VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

NÁVRH ŘÍDICÍHO SYSTÉMU PRO MALÝ ZKUŠEBNÍ STROJ

CONTROL SYSTEM DESIGN FOR SMALL TEST MACHINE

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. David Rasocha

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Stanislav Věchet, Ph.D.

SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Bc. David Rasocha**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**doc. Ing. Stanislav Věchet, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Návrh řídicího systému pro malý zkušební stroj

Stručná charakteristika problematiky úkolu:

Malý zkušební stroj je poháněn krokovým motorem s integrovanou řídicí jednotkou disponující sériovým komunikačním rozhraním doplněným o digitální vstupně/výstupní piny. Cílem práce je zvolit hardware řídicího systému, který bude umožňovat konfiguraci pohonu, řízení pohybu pohonu v závislosti na měřeném posuvu a komunikaci s nadřazeným softwarem. Software řídicího systému bude instalován na standardním laptopu a poskytne uživateli možnost nastavení, spuštění a analýzu měření.

Cíle diplomové práce:

1. Seznamte se s řídicí jednotkou krokového motoru dle dokumentace výrobce.
2. Definujte základní požadavky na řídicí systém z hlediska uživatele.
3. Navrhněte strukturu řídicího systému.
4. Vyberte adekvátní hardware pro řídicí systém.
5. Vytvořte software řešení.
6. Prakticky ověřte funkcionalitu celého systému.

Seznam doporučené literatury:

BALÁTĚ, J.: Technické prostředky automatického řízení. Praha, SNTL 1986.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Tato práce se zaměřuje na návrh malého zkušebního stroje pro měření pevnosti materiálů v tahu. Cílem je vybrat vhodný řídicí hardware pro pohonnou jednotku, která disponuje sériovým komunikačním rozhraním. Pohon je realizován pomocí krokového motoru, který má mikrokrokování. Řízení motoru zajistí kontrolér, který komunikuje s aplikací v počítači. Tato aplikace má všechny funkce nutné k plnohodnotnému řízení stroje.

Abstract

This thesis focuses on design of small testing machine for measuring tensile strength of materials. Appropriate hardware for driving the motor with serial communication will be used. Main drive is a stepper motor with microstepping. Instructions for motor is provided by microcontroler which will be comunicating with aplication in computer. This aplication will have all user functions nessesary for using this device.

Klíčová slova

Zkušební stroj, tahová zkouška, Jazyk C, Python,

Key words

Testing machine, tensile test, language C, Python

Bibliografická citace

RASOCHA, David. *Návrh řídicího systému pro malý zkušební stroj.* Brno, 2020. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/124895. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Stanislav Věchet.

Čestné prohlášení

Prohlašuji, že diplomovou práci „Návrh řídicího systému pro malý zkušební stroj“ jsem zpracoval sám za pomoci mého vedoucího práce a odborné literatury kterou jsem uvedl v seznamu zdrojů.

V Brně dne …….. ……

podpis

Poděkování

Tímto bych rád poděkoval doc. Ing. Stanislavu Věchetovi Ph.D. za pomoc a rady při tvorbě této práce.

Obsah

1 Úvod

Mechanické vlastnosti materiálů jsou stěžejní při návrhu konstrukcí, strojů a dalších technických zařízení. Tyto vlastnosti se převážně určují destruktivními zkouškami. Při destruktivních zkouškách dochází k porušení či zničení testovaného vzorku. Nejběžnější takovou zkouškou je zkouška tahem, při níž dochází k roztržení vzorku vlivem tahové síly. Dále jsou zkoušky v ohybu, krutu, nebo střihu. Zkoušky se dají rozdělit dále dle typu namáhání na dynamické, statické a cyklické. Cyklické zatěžování probíhá v dlouhém časovém úseku zatěžování silou v předem stanovených mezích. Při dynamickém zatěžování se vzorek zatíží náhle na předem definovanou mez. Statické zatěžování znamená, že se zatížení mění pomalu.

Zkušební stroje se většinou vyrábějí v různých velikostech. Velké stroje jsou konstruovány řádově na zatížení v tisících kilonewtonů. Malé jsou na zatížení od jednotek do stovek newtonů. Menší zařízení se hodí na trhání plastů nebo velice malých kovových vzorků.

Cílem této práce je navrhnout zařízení, které bude schopno tuto zkoušku provést. Toto zařízení má být malé, aby se mohlo umístit na pracovní stůl. Pro účely práce byl dodán rám se dvěma sloupy na pojezd čelistí. K rámu byl dodán také motor s vlastní řídicí jednotkou. Cílem je vybrat adekvátní elektroniku, aby bylo možná zařízení ovládat v zamýšleném rozsahu. Hlavním bodem je vytvořit softwarové řešení. Toto řešení by mělo umožňovat plně zařízení ovládat a zároveň by mělo být dostatečně intuitivní, aby ho mohla obsluhovat nezasvěcená osoba.

2 Požadavky

Výstupem této práce by mělo být grafické uživatelské prostředí, které bude dovolovat uživateli plně řídit zkušební stroj.

Variabilita rychlosti posuvu: Dvojí režim 1. rychloposuv 2. měření. Rychlost posuvu při měření bude defaultně nastavena na 5 mm/min nicméně uživatel musí mít možnost tuto rychlost měnit. Rychloposuv bude mít nastavenu rychlost na 20 mm/min ale také musí být změnitelná.

Měřicí režimy: Software uživateli musí umožnit nastavit si sílu předpnutí vzorku před samotným měřením. V samotném měření je požadováno, aby si uživatel mohl zvolit, jestli chce zařízení zastavit při konkrétním zatížení, nebo až po uražení konkrétní vzdálenosti.

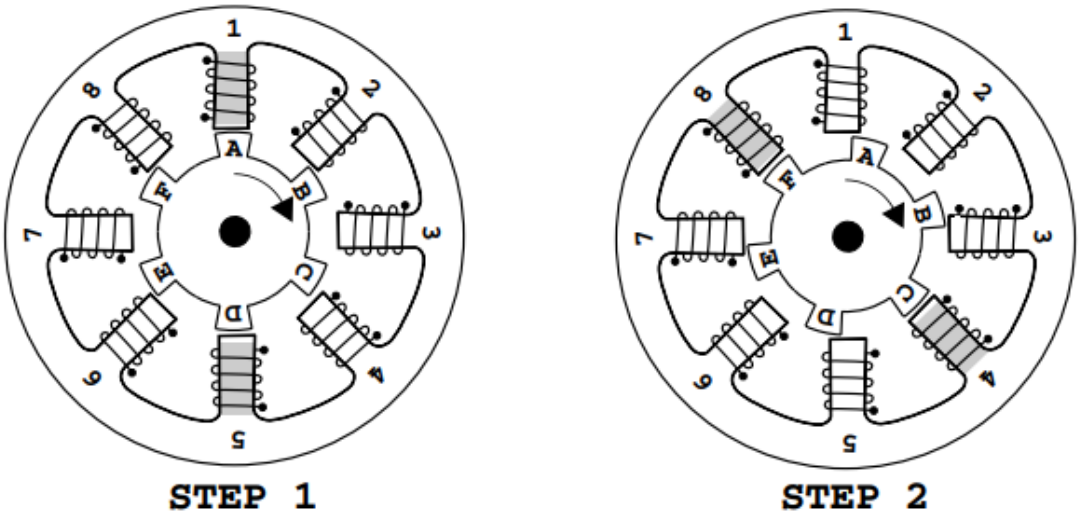
Data: Software musí umět zobrazit naměřená data. V datech musí být vektor síly, posuvu a času. Data se dají ukládat do textových formátů, tak aby mohla být načtena softwarem třetích stran.

Komunikace s aplikací Alfa: Musí být vytvořeno spojení mezi touto aplikací a softwarem třetí strany. Programy mezi sebou musí být schopné sdílet data v reálném čase.

3 Hardware

3.1 Krokový motor a mikrokrokování

Krokový motor díky své konstrukci umožňuje přesné řízení na polohu (úhel natočení) a i na otáčky. Motor převádí elektrické pulsy na otočení o určitý úhel, ten je zpravidla dán vnitřní konstrukcí motoru.



Obrázek 1: Princip činnosti krokového motoru [1]

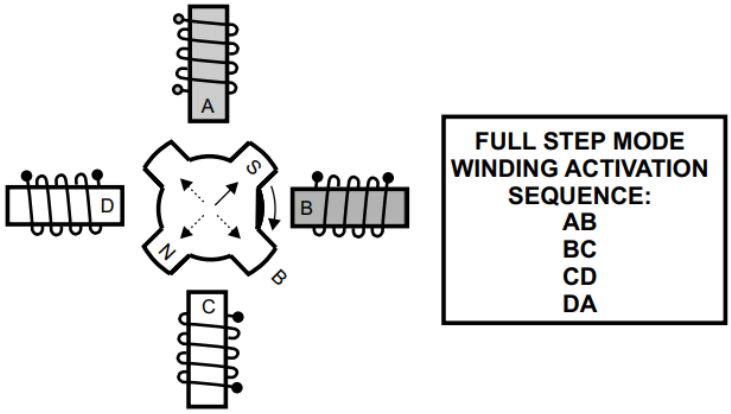
Na obrázku 1 je příčný průřez krokovým motorem. Rotor motoru má tzv. zuby a může být vyrobený z feromagnetického materiálu (reluktanční motor) nebo může být opatřen permanentními magnety. Vlastností krokového motoru s permanentními magnety je, že i při vypnutém napájení generuje neustále statický moment na hřídeli, tedy se „brání proti otáčení“. Nevýhodou je, že motor s magnety je vždy citlivý na teplotu, která je dána konkrétním typem použitých magnetů. Pokud je překročena tzv. Curieova teplota, ztrácí magnety své vlastnosti a motor je zničen.

3.1.1.1 Řízení krokového motoru

Oproti stejnosměrnému motoru se krokový motor nedá řídit velikostí napětí, ale spínám jednotlivých cívek statoru. K tomu se využívá mikroprocesorů. Dále se rozlišuje, jestli je řízení na takzvaný fullstep, halfstep nebo mikrokrokování.

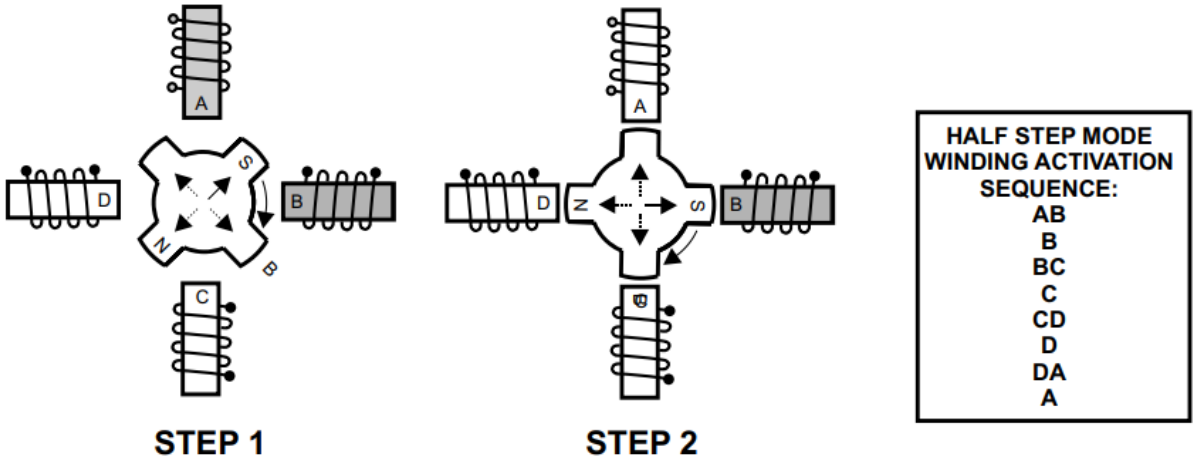
Fullstep a halfstep

Jak je již z názvu patné, jde o řízení, při kterém motor vykoná na jeden řídicí impuls celý krok. Obrázek 3 znázorňuje, jak jsou v takovém případě spínány cívky statoru.



Obrázek 2: Full step [1]

Halfstep umožňuje zdvojnásobit počet kroků motoru na jednu otáčku bez nutnosti změnit konstrukci motoru.



Obrázek 3: Halfstep [1]

Mikrokrokování

Režim mikrokrokování umožňuje zvětšit počet kroků na otáčku pomocí speciálního řízení napájení cívek. U běžného řízení na celý nebo poloviční krok je vždy do jedné nebo více cívek puštěno maximální napájecí napětí a tím je vytvořen maximální magnetický tok. Při režimu mikrokrokování je napětí na cívkách měněno v závislosti na požadované poloze rotoru. Tento typ řízení snižuje maximální moment motoru, nicméně dochází k vyhlazení momentu v závislosti na volbě počtu kroků.

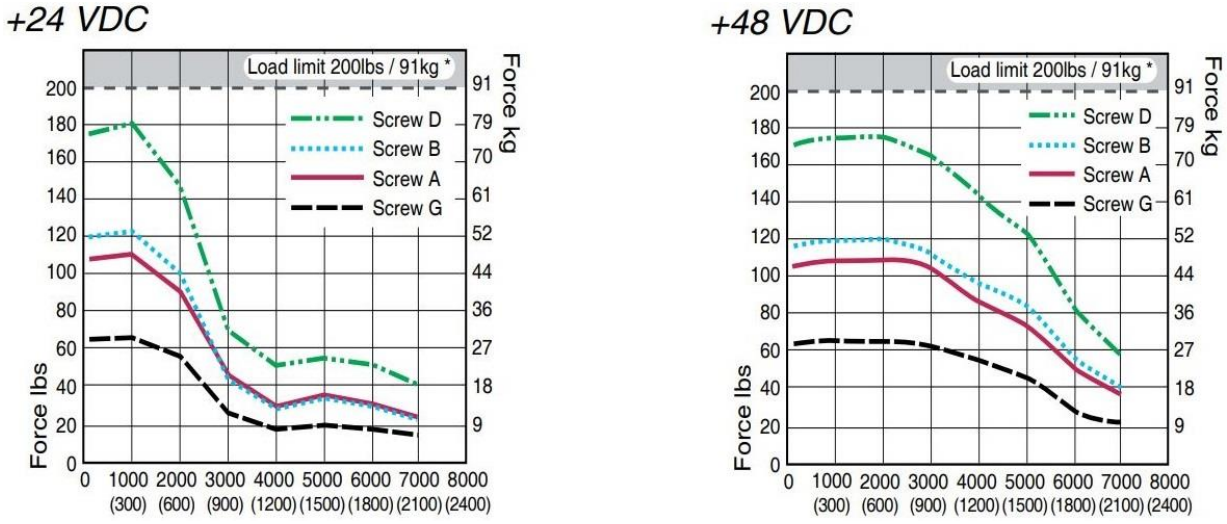
Aby nedocházelo k výraznému poklesu momentu motoru, tak se používá řízení v uzavřené smyčce.

3.2 Použitý motor

Stroj je poháněn krokovým motorem MDrive23Plus. Jedná se o dvoufázový krokový motor s mikrokrokováním, který má základní úhel kroku 1,8°. Osou motoru prochází šroub, který funguje jako přenašeč síly a rotor je umístěn okolo něj. Výrobce nabízí několik typů šroubu, které se liší stoupáním závitu. Použitý šroub má stoupání 2,116 mm na otáčku. Motor dále obsahuje vlastní řídicí jednotku a vstupně výstupní programovatelné piny.

Obrázek 4: Použitý krokový motor [2]

Napájení motoru má dle výrobce rozsah 12 až 75 V stejnosměrně. Velikost napájecího napětí přímo ovlivňuje momentovou křivku motoru. Odebíraný proud je maximálně 2 A při 12 V.



Obrázek 5: Silové křivky motoru pro různé velikosti napájení [2]

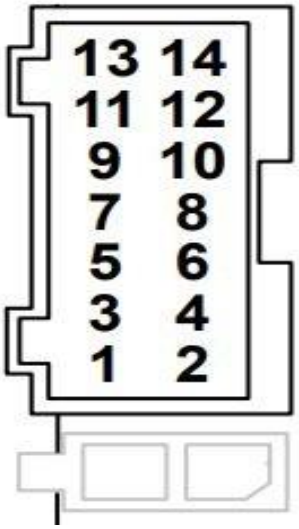
Na obrázku č 5 jsou silové charakteristiky motoru (jedná se o sílu kterou dokáže vyvinout šroub podél své osy). Na vodorovné ose jsou kroky za minutu (jeden krok odpovídá pootočené o 1.8°), v závorce jsou otáčky za minutu. Moment, který produkuje motor posouvá šroub a síla, kterou je tímto způsobem možné vytvořit je závislá také na stoupání závitu šroubu. Tento vliv

lze pozorovat na obrázku 5. Pro naše účely je použit „screw D“, tedy šroub s nejmenším stoupáním. Maximální tah udávaný výrobcem je 79 Kg.

3.3 Řídicí jednotka motoru a komunikace

Motor obsahuje vlastní řídicí jednotku, která má univerzální vstupní port. Tento port obsahuje piny pro napájení motoru a také pro SPI připojení. Jedná se o čtrnácti pinový konektor, rozložení pinu na obrázku 6. Konektor se nachází zboku motoru.

Tabulka 1: Vstupně výstupní piny motoru

Nákres konektoru [2]	**Číslo pinu**	**Popis funkce**
	1	I/O Kladný pól
	2	I/O Záporný pól (GND)
	3	I/O programovatelný pin 1
	4	I/O programovatelný pin 2
	5	I/O programovatelný pin 3
	6	I/O programovatelný pin 4
	7	I/O programovatelný pin 5
	8	I/O programovatelný pin 6
	9	I/O programovatelný pin 7
	10	I/O programovatelný pin 8
	11	Capture pin
	12	Analogový vstup 0 až 10 V
	13	Digitální rychlostní pin
	14	Digitální směrový pin (DIR)

Všechny programovatelné piny konektoru pracují v napěťové úrovni 0 až +24 V. Pin číslo 12 je analogový vstup s omezením na 10 voltů. Piny 13 a 14 jsou pro přímé ovládání motoru, čtrnáctý pin udává směr otáčení – pokud je v logické nule, motor se otáčí po směru hodinových ručiček. Třináctý pin slouží pro ovládání otáčení, každá náběžná hrana znamená pootočení o jeden krok. Poslední dva piny pracují v TTL logice, která definuje logickou jedna od úrovně 2 do 5 voltů. Logická nula je definována od 0 do 0,8 V. Pásmo od 0,8 do 2 V není standardem definováno, a proto se vstupní signál nesmí v této úrovni pohybovat.

Jednotka motoru je nastavena tak aby bylo 256 mikrokroků na 1.8° pootočení, tedy 51200 kroků na jednu otáčku. Rychlost posuvu v závislosti na frekvenci řídicích pulzů lze popsat vztahem:

$$v = stoupani \cdot \frac{f}{51200}$$

Kde stoupání je 2,116 mm na otáčku a f je frekvence řídicích pulsů. Minimální šířka řídicího pulsu je dle manuálu 100 ns. Maximální frekvence, kterou je kontrolér motoru zachytit je 10 MHz

Vedle datového konektoru na motoru se ještě nachází konektor napájecí, který je oddělený a podporuje napájení od 12 do 60 V. Toto napájení slouží i pro integrovanou jednotu motoru. Maximální odběr je dle výrobce 2 A.

3.4 Měření síly

K měření síly se v inženýrské praxi běžně používají odporové tenzometry. Využívá se při tom deformace součásti, na které je tenzometr umístěn. V podstatě lze tenzometry využít na měření deformace součásti ať už vlivem mechanického zatížení či teploty, nebo na měření síly působící v určitém místě a to tak, že se tenzometr umístí na speciální kus materiálu, který má známou geometrii a materiálové vlastnosti podle kterých lze z deformace dopočítat působící sílu. Dalším důležitým vlivem je způsob zatěžování, rozlišujeme: tah, tlak, ohyb, krut.

Odporové tenzometry pracují na principu změny elektrického odporu vodiče se změnou jeho geometrie. Tento jev je popsán rovnicí:

$$R = \sigma \cdot \frac{l}{S}$$

Kde:

R – odpor vodiče
σ – měrný elektrický odpor materiálu
l – délka vodiče
S – průřez vodiče

V tenzometrii se používá takzvaná poměrná změna odporu, která se zapíše jako:

$$\frac{\Delta R}{R} = \varepsilon \cdot k$$

Proměnná ε je přetvoření a k je takzvaný Součinitel deformační citlivosti též uváděný jako K-factor. Z výše uvedené rovnice se vyjádří ε a pomocí Hookeova zákona se dopočítá přímo síla nebo moment působící na vzorek. Vlákno materiálu musí být z materiálu, který vykazuje co nejmenší změnu odporu při změně jeho teploty. Dále musí být lineárně elastický, aby na něj mohl být aplikován Hookeův zákon. Hlavní rušivé vlivy, které se musí kompenzovat jsou teplota a také odpor vodičů vedoucích do tenzometru.

Tenzometry se dají koupit od výrobce ve více provedeních podle způsobu zatěžování a podle aplikace. Nejčastěji se vyskytují tenzometry fóliové, které se nalepí na součást a nejsou opětovně demontovatelné. Pro měření zatížení, síly atd. se často využívají deformační členy. Takový člen obsahuje tenzometry a je připravený pro aplikaci, opět je rozhodující způsob zatěžování a rozsah ve kterém lze člen používat, aby nebyl poškozen.

Pro tuto aplikaci byl použit deformační „S“ člen z obrázku 6 od firmy HBM. Tento člen slouží pro měření tahu nebo tlaku v mezích 10 až 1000 N. Je zapojen do plného mostu s kompenzací odporu vodičů a stíněním proti rušivému elektromagnetickému šumu z okolí.

Obrázek 6: Deformační "S" člen pro tah-tlak [3]

Napájení tenzometrů bude využit měřící zesilovač od HBM. Výstup ze zesilovače má hodnoty -2,5 až 2,5 V, kladné napětí znamená tah.

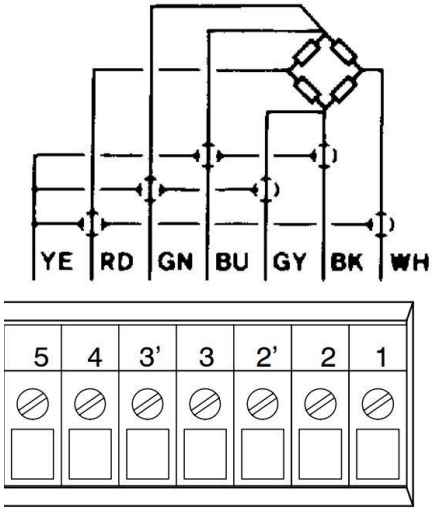
3.4.1 Tenzometrický zesilovač

Samotný tenzometr je třeba napájet a zároveň měřit rozvážení mostu. K tomu se hodí Tenzometrický zesilovač od stejnojmenné firmy jako je deformační S člen. Pro tuto práci byl použit model AE101, který měří tenzometry v zapojení plného mostu s kompenzací vedení.

Type	**AE101, AE301, AE501**	
Terminal	**Function**	**Color (HBM - cable)**
1	Measuring signal	WH (white)
2	Bridge excitation voltage	BK (black)
2'	Sensor line	GY (grey)
3	Bridge excitation voltage	BU (blue)
3'	Sensor line	GN (green)
4	Measuring signal	RD (red)
5	Screen/Ground	YE (yellow)
8	Synchronization (not with AE101)	
9	Operating-voltage zero*)	
10	Output voltage	
11	Supply voltage zero*)	
12	Supply voltage	

Obrázek 7: Tabulka zapojení pinu zesilovače [4]

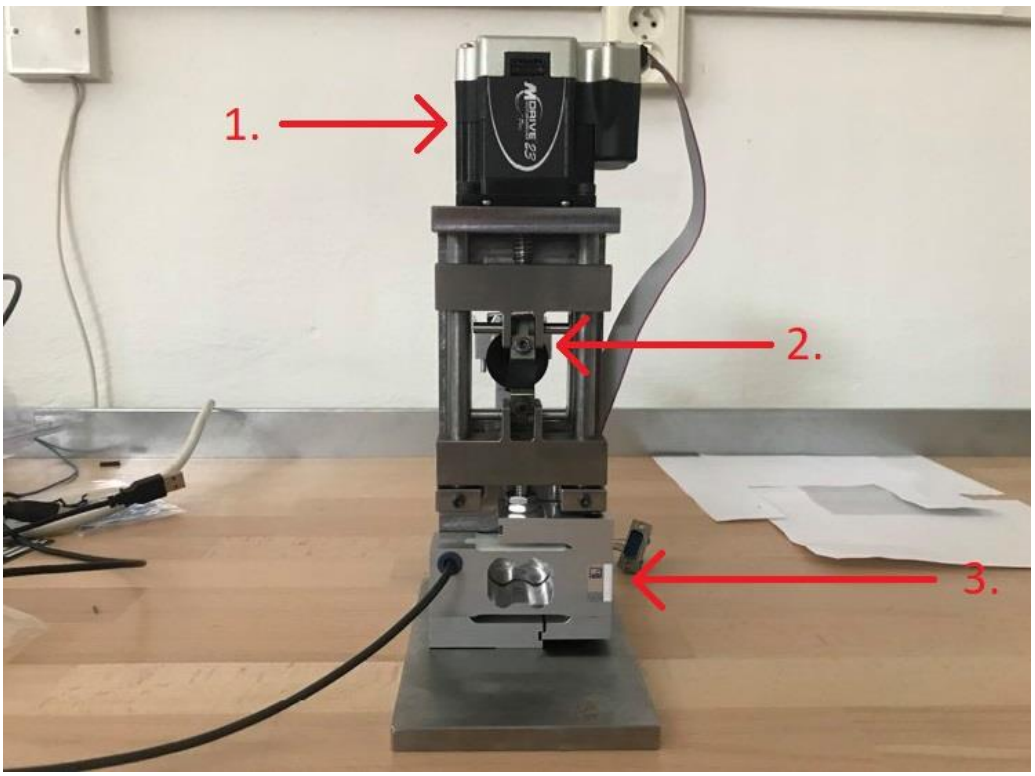
Napájení měřící stanice je výrobcem udáváno v rozmezí 15 až 26 V stejnosměrně. Výstupní napětí je vždy symetrické okolo nuly a je možné ho nastavit na tři napěťové úrovně. Pro účely této práce byl zvolen rozsah +-2,5 V. Dále jsou uvnitř přepínače, pomocí kterých se kalibruje pro konkrétní tenzometr. V manuálu výrobce se nachází tabulka, která říká, jak nastavit přepínače v závislosti na napětí a milivoltech na volt. Když je tato procedura hotová, je třeba vynulovat měřicí stanici. To se provede ve dvou krocích, prvně je třeba odpojit tenzometr, poté připojit na výstup voltmetr a pomocí otočného vypínače S23 zkalibrovat na nulové napětí.



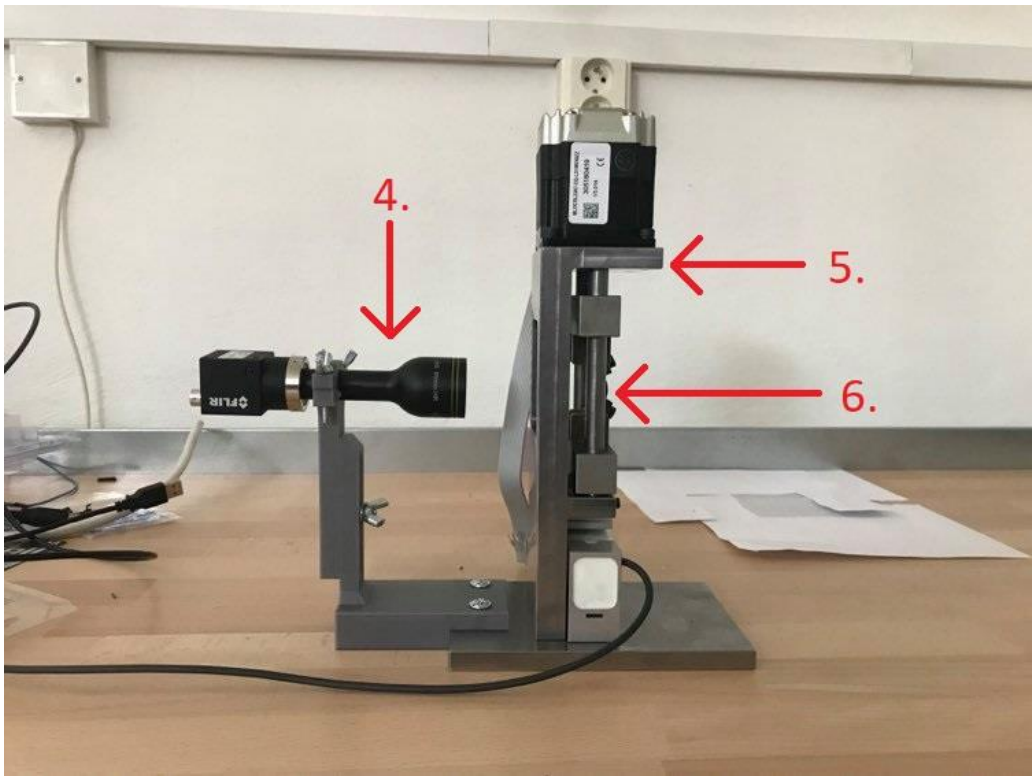
Obrázek 8: Zapojení plného mostu do zesilovače [4]

3.5 Konstrukce

V rámci této diplomové práce byl navrhnut řídicí systém pro zkušební stroj, ke kterému byla dodána konstrukce. Popis konstrukce je na obrázku číslo 9 a 10. Konstrukce je velmi robustní, aby nedocházelo k ohýbání jejích částí během trhání.



Obrázek 9: Trhačka zepředu



Obrázek 10: Trhačka z boku

Popis konstrukce a součástí trhačky:

1. Krokový motor
2. Čelisti pro upnutí vzorku
3. Tenzometrický siloměr
4. Kamera pro digitální obrazovou korelaci
5. Nosná konstrukce
6. Vodicí tyče čelistí

3.6 Kontrolér

Pro samotné ovládání motoru je třeba zvolit mikrokontroler který bude zajišťovat komunikaci s motorem a dále snímat sílu ze zesilovače. Mikrokontroler je elektronické programovatelné zařízení pro všestranné použití. Jelikož se jedná o celkově jednoduchou úlohu, není třeba požadovat vysoký výkon mikroprocesoru. Od kontroléru je vyžadováno, aby byl schopen komunikovat s počítačem a aby byl vybavený periferiemi, jako je například ADC převodník a také PWM modulací. Uvedené periferie bývají standardem u téměř jakéhokoliv mikroprocesoru. Z důvodu snadného používání a množství uživatelsky přívětivých funkcí, byla zvolena platforma Arduino.

Obrázek 11: Arduino Leonardo [5]

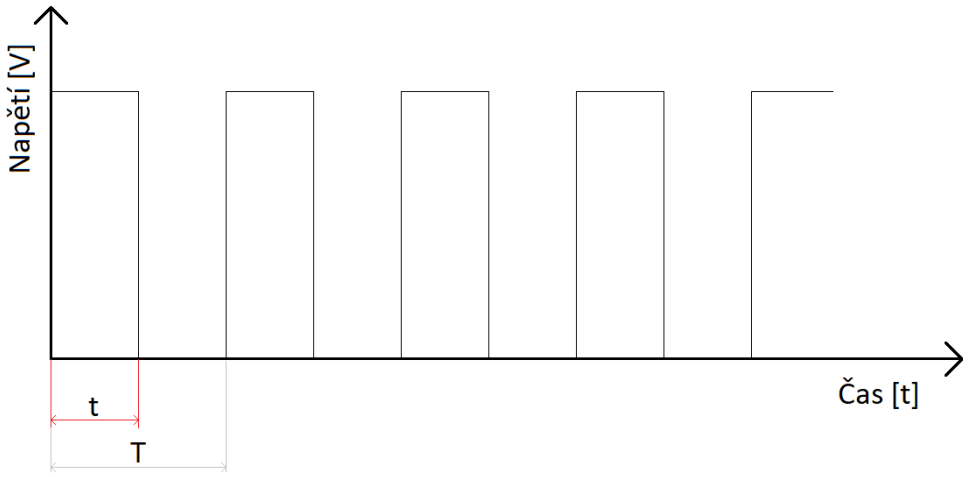
Kontroléry Arduino jsou vybaveny procesory od firmy Atmel, všechny procesory jsou osmibitové a je zachována kompatibilita kódu mezi jednotlivými deskami.

3.6.1 Periferie

Pulsně šířková modulace

Pulsně šířková modulace, též označována PWM slouží ke generování periodického analogového signálu, který nabývá pouze dvou logických úrovní. Nejčastější použití je pro řízení spínání tranzistoru v měničích, kde střída zároveň procentuálně odpovídá výstupnímu napětí měniče. Jak již bylo řečeno, signál nabývá pouze dvou úrovní a to typicky 0 a 5V nebo podle maximální voltáže napájení mikroprocesoru. Dalším charakteristickým parametrem PWM je střída, ta udává šíři pulsů. Střída je definována jako poměr času, kdy je hodnota v horní logické hladině ku periodě signálu (obrázek 9). Matematicky zapsáno takto:

$$S = \frac{t}{T}[-]$$



Obrázek 12: Pulsně šířková modulace v časové oblasti se střídou 50%

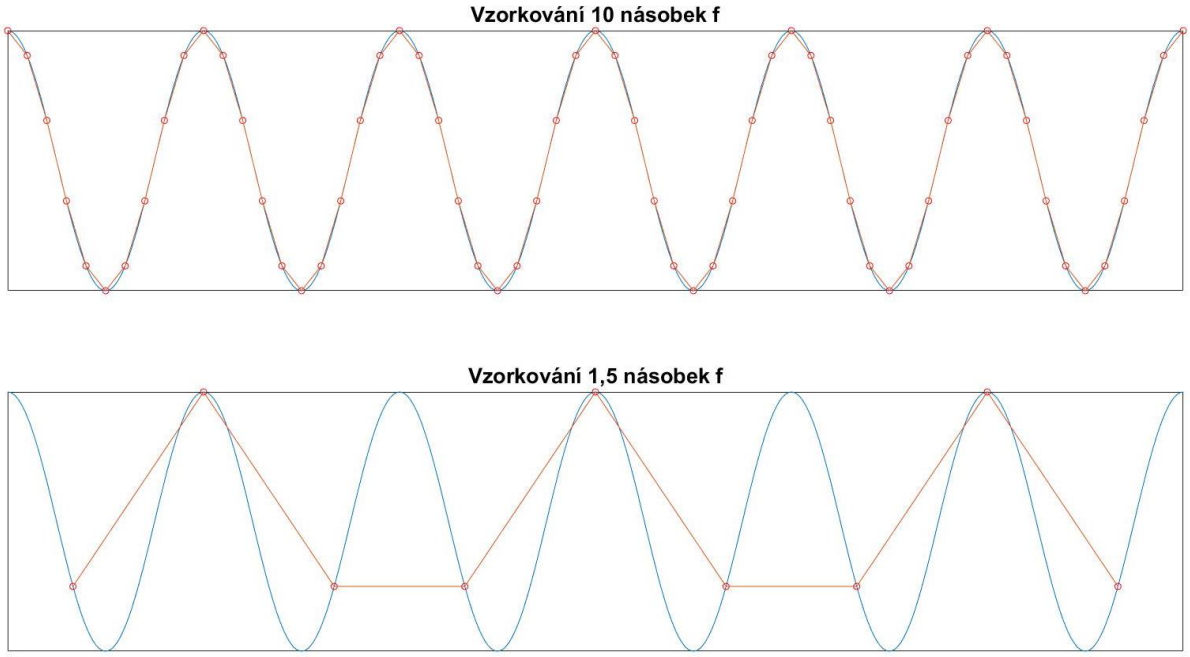
A/D převodník

Analogově digitální převodník (anglicky Analog to Digital Converter) je periferie procesoru, která slouží pro převod analogových elektrických veličin do paměti procesoru. Typicky výrobce dává do mikroprocesorů 10 nebo 12bitové převodníky. Počet bitů je směrodatný parametr, neboť určuje, kolik bitů bude mít číslo v paměti. Lze tedy říct, že do určité míry počet bitů odráží přesnost převodníku. Velikost čísla v decimálním zobrazení lze spočítat ze vztahu:

$$decimální = 2^{počet\ bitů}$$

Dalšími důležitými pojmy jsou vzorkovaní a kvantování, abychom pochopili funkci převodníku je nutné znát význam těchto pojmů. Převodníky snímají libovolnou analogovou veličinu, kterou si musí vhodně rozdělit – vzorkování.

Hlavní a nejdůležitější věcí u vzorkování je takzvaná vzorkovací frekvence, tedy číslo, které říká kolikrát se zaznamená veličina během jedné sekundy. Pokud se vzorkuje frekvencí větší, než je nutné, tak nevzniká žádný problém, nicméně v opačném případě může dojít ke zkreslení signálu. To znamená, že naměřená data neodpovídají skutečnému signálu. Tento problém popisuje Nyquistův teorém, který říká, že vzorkovací frekvence musí být aspoň desetkrát větší, než je nejvyšší frekvence v zaznamenávaném signálu, aby byl signál zcela rekonstruován.



Obrázek 13: Vliv vzorkování, f= 60 Hz

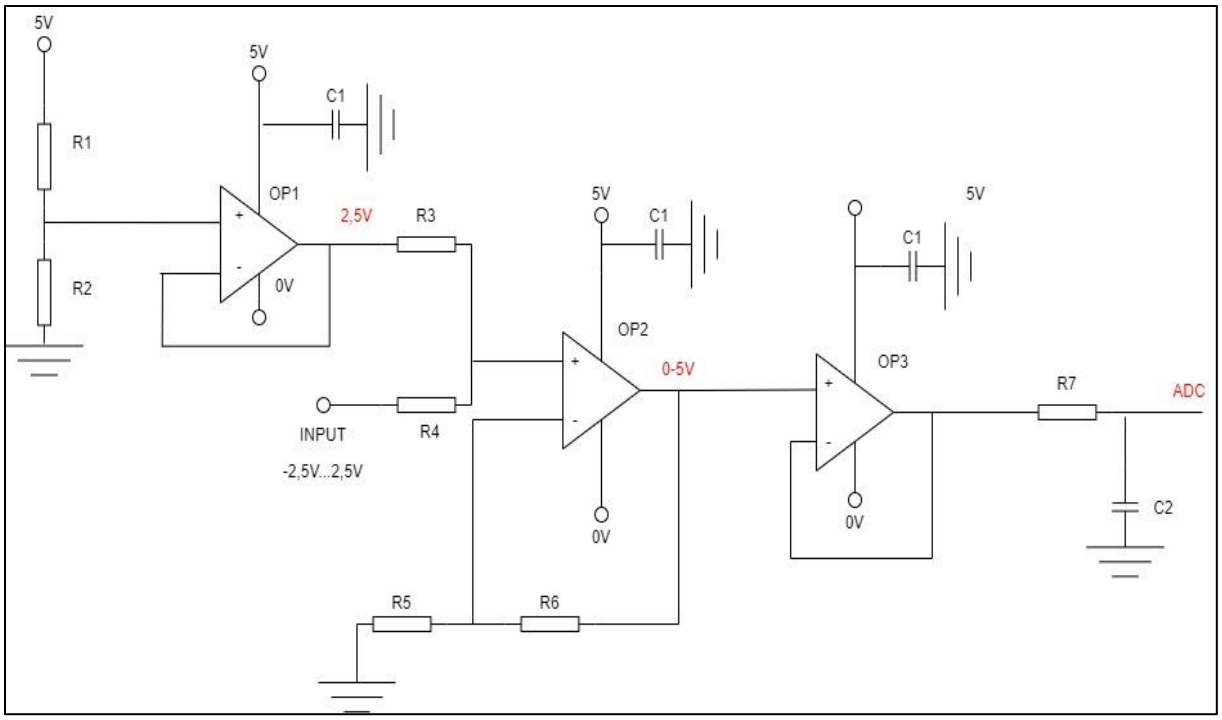
Obrázek 13 demonstruje vliv vzorkovací frekvence na nasbíraná data. V prvním případě je vzorkování přesně desetinásobek frekvence signálu, ve druhém případě je vzorkovací frekvence 1,5násobek frekvence signálu. Červeně jsou zvýrazněna měřená data.

Kvantování je děj, při kterém se přiřazuje číslicová hodnota signálu. Pásmo, ve kterém je signál snímá je rozděleno na 2^n dílků, kde „n“ je počet bitů převodníku. Dále je celý interval rozpůlen a pomocí komparátoru je určeno v jaké polovině se nachází měřená hodnota, tento děj se opakuje, dokud není nalezena úroveň odpovídající měřenému signálu. Je nutné dodat, že s narůstajícím počtem komparačních úrovní převodníků dramaticky klesá rychlost vzorkování.

Arduino disponuje 10bitovým ADC, které neposkytuje požadovanou přesnost, proto byl zakoupen externí převodník, který je 16bitový. Externí převodník je, jakou u Arduina, unipolární, není tedy možné zaznamenávat záporné hodnoty napětí. Referenční hodnota napětí je na stavena na 5V. Komunikace s kontrolérem probíhá přes I2Cstandard.

3.6.2 Napěťové přizpůsobení

Jelikož je kladen požadavek na měření zatížení v obou směrech, je nutné upravit signál z tenzometru. Výstup z tenzometru má rozsah -2,5 až 2,5 V nicméně je tento signál nutné upravit tak, aby byl měřitelný unipolárním převodníkem. Z těchto důvodů byl vytvořen jednoduchý elektronický obvod, který signál posune do mezí 0 až 5 V. Aby nedocházelo k úbytkům napětí na operačních zesilovačích, byly použity takzvané rail-to-rail zesilovače. Tyto zesilovače mají tu vlastnost, že na svůj výstup poskytnout téměř stejné napětí jako kterým jsou napájeny. Tento obvod je umístěný na oddělitelném plošném spoji i s externím ADC převodníkem. Schéma obvodu je na obrázku 14.



Obrázek 14: Schéma obvodu pro posun napětí

Schéma se skládá ze tří operačních zesilovačů. První zesilovač je v zapojení sledovače, to znamená že impedančně odděluje dělič napětí R1 a R2 aby nedocházelo k jeho rozvážení vlivem toku proudu a má zesílení rovné 1. Dělič R1, R2 dělí vstupní napětí na polovinu čímž vytváří referenci pro měřený signál. Poslední operační zesilovač je také v režimu „sledovače" aby nebyl rozvážen dělič R5 a R6. OP2 je v neinvertujícím zapojení, kde porovnává vstup s výstupem, který je redukován děličem. Kondenzátor C1 je umístěn na napájecí větev, aby stabilizoval případné záchvěvy napětí.

Víme, že vstup nabývá hodnot -2,5 až 2,5 V. Řekněme tedy že chceme využít celý interval ve kterém je možné měřit. Pak hodnota -2,5 V odpovídá na výstupu 0 V a +2,5 V odpovídá hodnotě +5 V. Z předešlého vyplívá, že rozpětí napětí je stejné a je nutné ho jen posunout o trvalý offset. Napětí neinvertujícího vstupu lze spočítat s využitím druhého Kirchohoffova zákonu. Matematický zápis:

$$U_{neivert} = 2{,}5 \cdot \frac{R3}{R4 + R3} + U_{vstup} \cdot \frac{R4}{R4 + R3}$$

Z této rovnice plyne, že napětí na neinvertujícím vstupu je v mezích 0 až 2,5 V. Abychom dosáhli požadovaného výstupu, je třeba zapojit OP tak aby měl dvojnásobné zesílení. Toho se dosáhne přidáním děliče R5 a R6. Napětí na výstupu OP2 je dáno rovnicí:

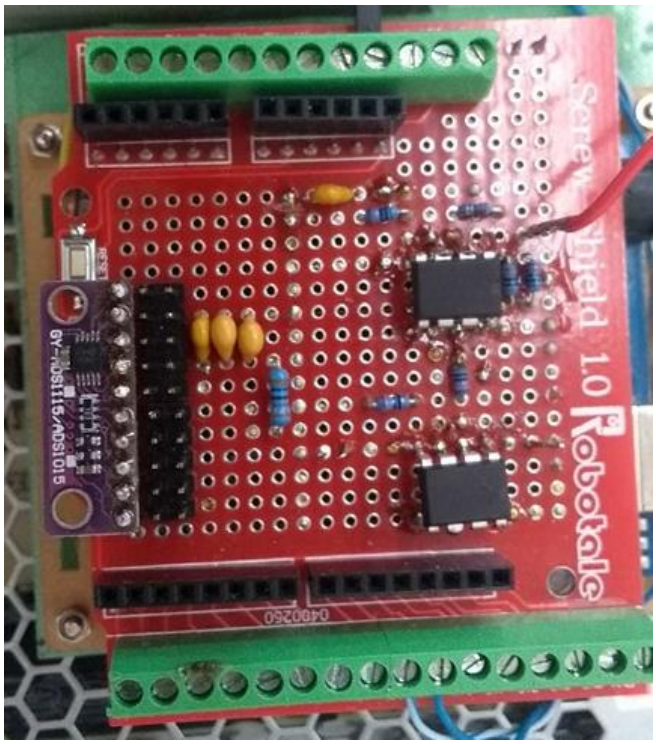
$$U_{výst} = U_{neinvert} \cdot (1 + \frac{R6}{R5})$$

Hodnoty odporů R1 až R6 v zapojení jsou 10 kΩ protože jsou to vždy děliče na poloviční napětí. Velikost filtračního kondenzátoru C1 byla zvolena 100 nF. Kondenzátor je keramický.

Na konci se nachází kombinace odporu R7 a kondenzátoru C2. Toto zapojení představuje antialiasingový filtr. Ten má za úkol odfiltrovat všechny vyšší frekvence. Filtr má podobu dolní propusti. Mezní hodnota propusti se spočítá pomocí vzorce:

$$f = \frac{1}{2 \cdot \pi \cdot R \cdot C}$$

Pro účel této práce bych zvolen odpor 1kΩ a kapacita kondenzátoru 300 nF. Při této kombinaci jsou potlačovány frekvence vyšší než 530 Hz. To je dostatečné pro odfiltrování vysokofrekvenčního šumu, který se jinak v obvodu vyskytuje. Zapojení v praktickém provedení je na obrázku 15.



Obrázek 15: Elektronika ve finálním provedení

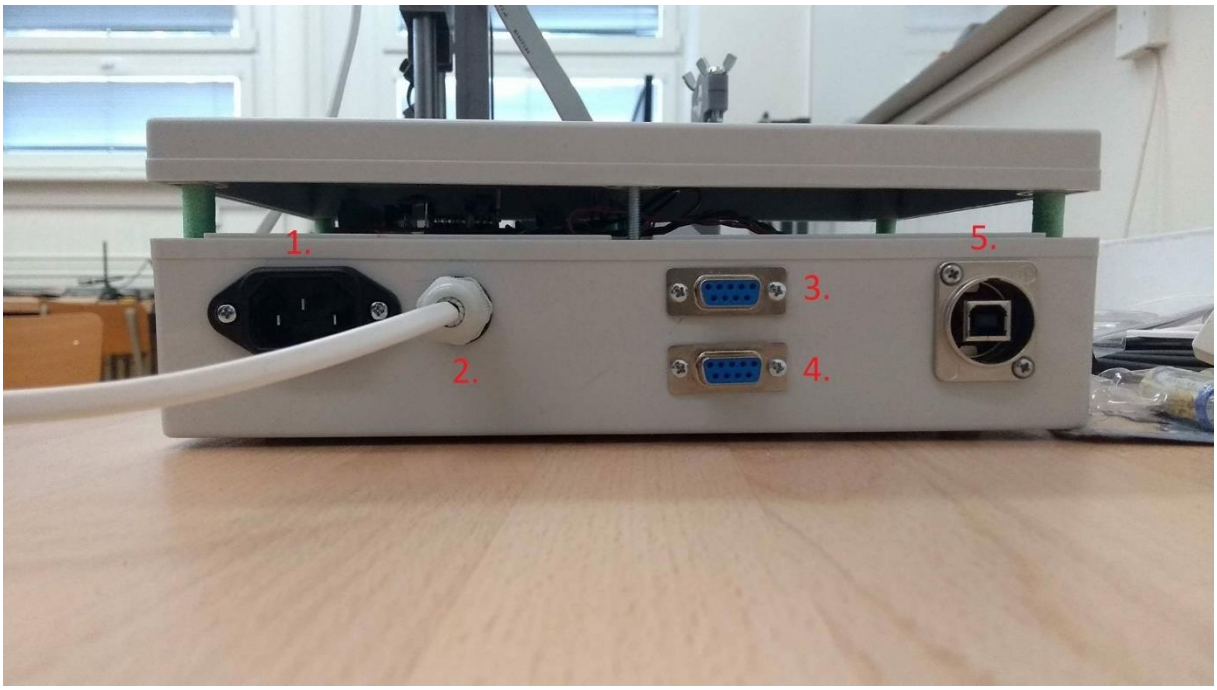
3.7 Napájecí zdroj

Pro napájení všech komponent (motoru, tenzometrického zesilovače) je použit externí zdroj který má vstup na síťové napětí 230 V. a výstupem je 15 V stejnosměrného napětí. Je to spínaný zdroj s vysokou účinností v zatížení a malými ztrátami v režimu, kdy neposkytuje žádný výkon. Maximální výkon, který je schopen dodávat po dlouhou dobu je 100 W. Zdroj má ochranu proti zkratu. Použitý zdroj je na obrázku 16.

Obrázek 16: Napájecí zdroj [6]

3.8 Sestava

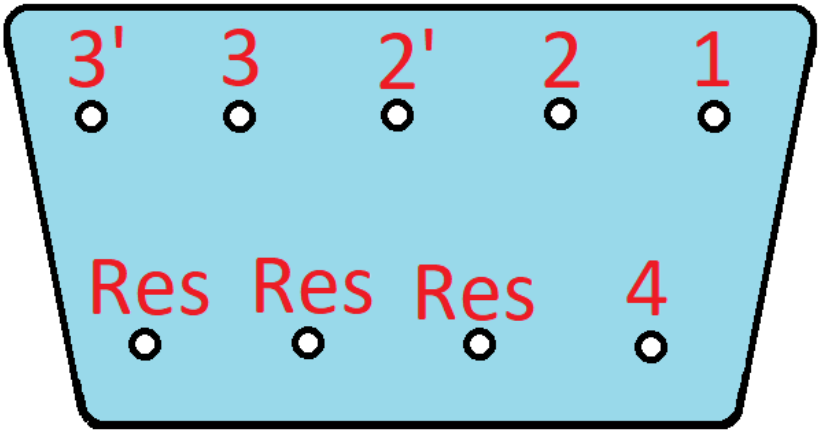
Veškerá elektronika byla umístěna do boxu. Box po své straně obsahuje konektory pro připojení napájení a datové konektory, jak je na obrázku 15. Aby nedocházelo k zbytečnému přehřívání komponent uvnitř boxu, bylo víko uloženo na distanční podložky.



Obrázek 17: Přípojka k elektronice

1. Připojení napájení 230 V
2. Napájení motoru 15 V
3. CAN konektor pro připojení motoru
4. CAN konektor pro připojení tenzometru
5. USB typu „B“ pro připojení počítače

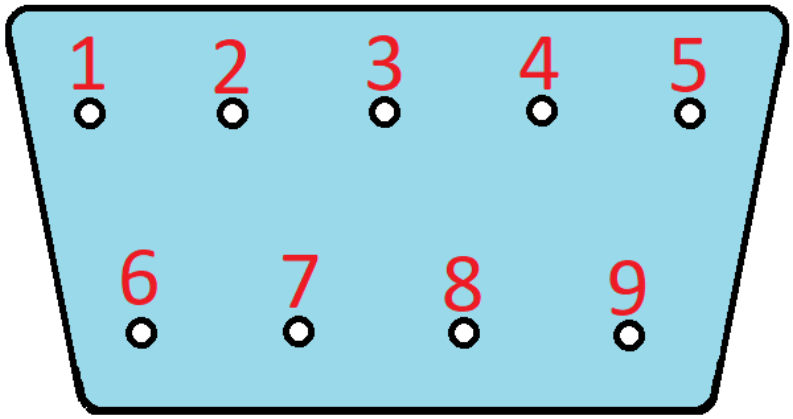
3.8.1 Pinout CAN konektorů



Obrázek 18: Číslování pinů CAN konektoru

Konektor tenzometru: Připojení tenzometru je realizováno pomocí CAN konektoru. Značení pinů na obrázku 18 odpovídá značení na obrázku 8. Piny označené jako „Res“ jsou rezervované.

Konektor motoru: Pin číslo 2 je připojený na výstupní pin Arduina číslo 8. Tento pin je směrový. Pin číslo 3 je připojený na výstupní pin kontroléru číslo 9 a jsou na něm řídicí pulsy motoru. Číslování konektoru je na obrázku 19.



Obrázek 19: Konektor k motoru

3.8.2 Sestava uvnitř rozvaděče

Komponenty byly umístěny do rozvaděče. Finální provedení je na obrázku 20.

Obrázek 20: Rozložení komponent v rozvaděči

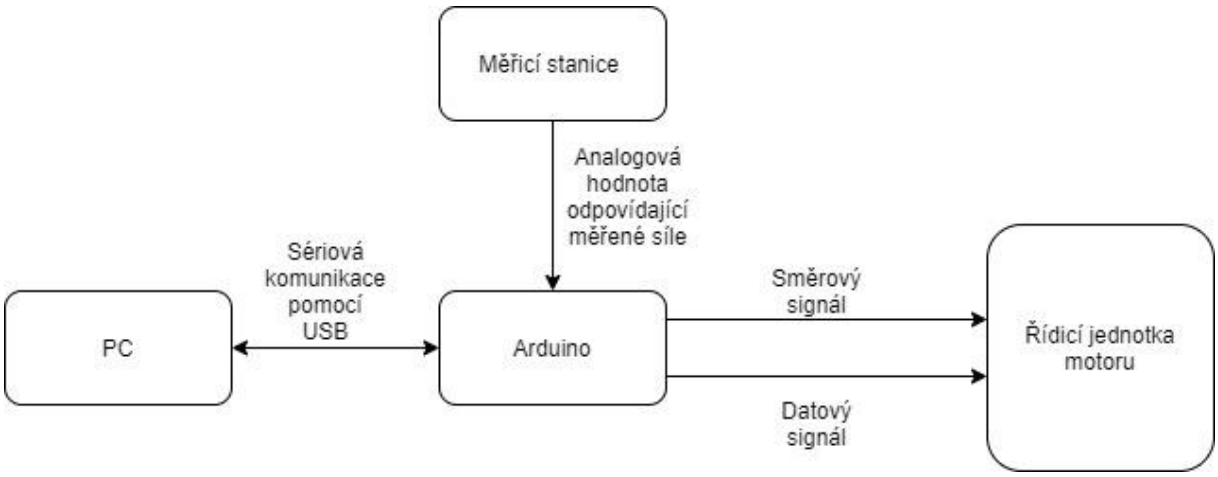
4 Software

Software lze chápat jako sadu instrukcí, které vykonají určitou činnost zamýšlenou autorem programu. Pro vývoj softwaru se používají programovací jazyky, kterých je celá řada. Hlavním kritérium pro výběr jazyka je jeho platforma, tedy cílové zařízení, na kterém program bude fungovat. Pro programování mikrokontrolerů je zpravidla realizováno v jazyku C. Tento jazyk poskytuje možnost používat proměnné a funkce jako vysokoúrovňové jazyky ale zároveň poskytuje možnost přímé správy paměti zařízení. Pro programování uživatelských prostředí - GUI (z anglického Graphic User Interface) se používají vyšší programovací jazyky hlavně z důvodu velké podpory knihoven které práci značně usnadňují.

Jelikož bude zařízení ovládáno z aplikace v počítači, je nutné stanovit hierarchii mezi softwarem. Nadřízený software musí být schopen v dostatečném rozsahu ovládat software podřízený. V praxi to znamená, že jeden ze softwarů je takzvaný „Master“ a druhý „Slave“. Master vždy vytváří akci, tedy slave nekoná nic z vlastní vůle, pokud tomu není pokyn od nadřízeného softwaru. V tomto případě je aplikace v PC master, jelikož ji obsluhuje uživatel, který musím mít kontrolu nad strojem.

4.1 Návrh řídicího sw mikrokontroleru

Mikrokontroler má za funkci řídit motor a zároveň vyčítat data o síle z tenzometru a posílat je do osobního počítače. Následující obrázek schematicky znázorňuje propojení Arduina s ostatními komponentami.



Obrázek 21: Schématické zapojení mikrokontroleru

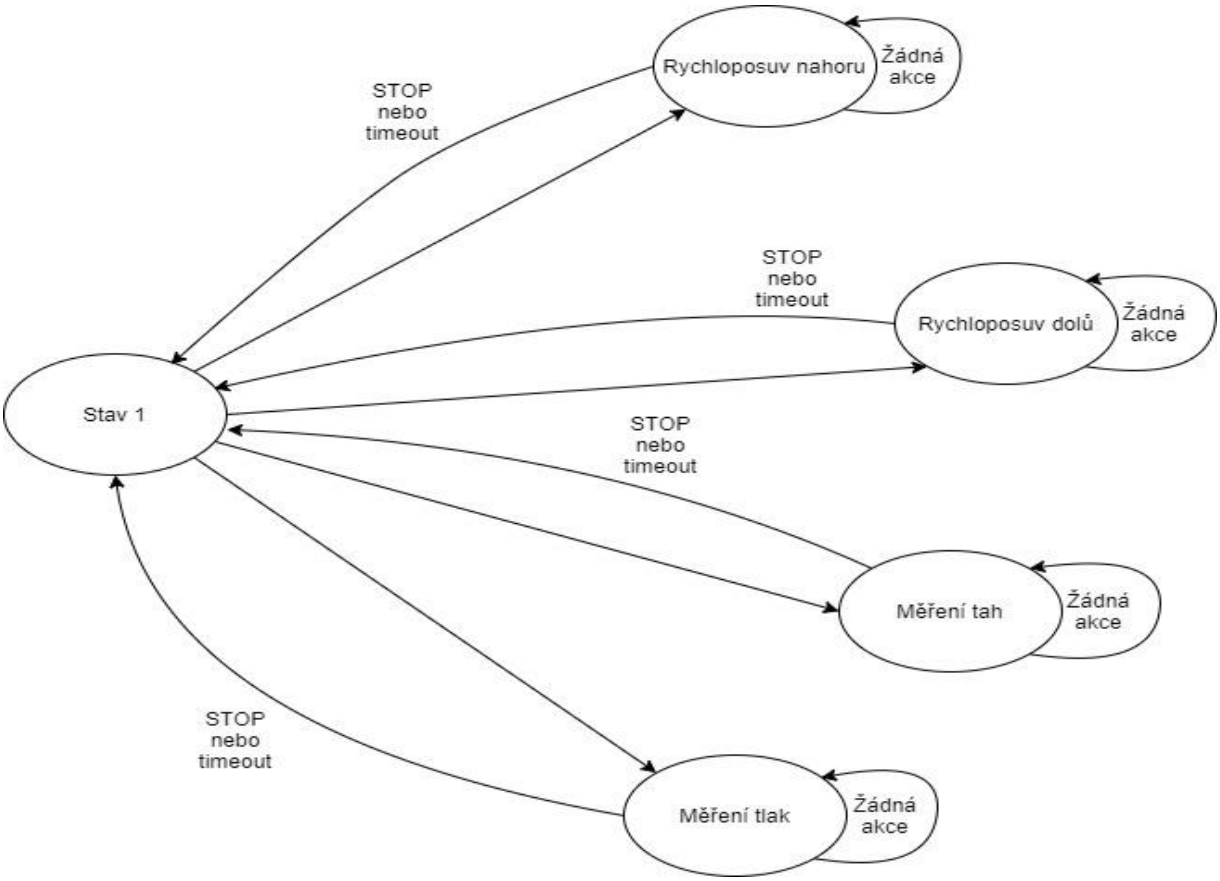
Jelikož procesor mikrokontroleru není schopen paralelně zpracovávat více než jednu úlohu, je třeba obsluhu všech komponent dělat ve smyčce. Dále je třeba aby program pracoval ve stavech automaticky dokud jeho stav není uživatelem změněn

4.1.1 Stavový automat

Stavový automat je nejběžnějším příkladem softwaru, který řídí nějaký automat či zařízení. Výhodou této struktury řízení je, že je lehce přenositelné do kódu. Další nespornou výhodou je, že je lehký na pochopení, a tak přispívá k čitelnosti samotného programu. Stavový automat je v podstatě soubor stavů, do kterých se program může dostat. Jednotlivé stavy mezi sebou

mohou mít určité vazby, a tak není vždy možné se dostat jakéhokoliv stavu do jiného. Běžně je jeden ze stavů takový, že pouze udržuje program v pohotovostním režimu, popřípadě zajišťuje nějakou základní funkcionalitu. Takový stav ve většině případů slouží jako výchozí stav.

V případě tohoto projektu bude využit stavový automat, který má pět stavů, které obsluhují chod stroje. Do jednotlivých stavů se program dostává pomocí příkazů, které posílá nadřazený software z počítače. V každém stavu program setrvává, dokud není přijat „STOP“ příkaz, který navrátí opět do počátečního „stavu 1“.



Obrázek 22: Diagram stavového automatu

4.1.1.1 Popis stavů

Stav 1: V tomto stavu se ocitne software hned po připojení ke zdroji energie. Tento stav je takzvaný počáteční stav. Software v tomto stavu čeká na připojení k počítači a poté automaticky začne vysílat pravidelně zprávy s hodnotou z ADC. Zároveň se kontroluje, zdali uživatel neposlal nějaký příkaz.

Rychloposuvy: Rychloposuv slouží k rychlému přesunutí čelistí stroje na místo. V tomto stavu je nastavena vysoká rychlost posuvu. Tato rychlost je změnitelná uživatelem z PC. Rychloposuv nahoru a dolů se liší pouze změnou napětí na směrovém pinu. Oba tyto stavy jsou aktivní, dokud nepřijde „STOP“ příkaz z počítače. Pokud přijde stop, software se opět vrací do počátečního stavu. I během rychloposuvu je měřena síla a posílána po sériové lince do počítače.

Měření: Během měření je nastaven velmi malý posuv. Opět jako u rychloposuvu je rozdíl mezi měřením tlaku a tahu pouze v napětí na směrovém pinu. Základní rychlost posuvu je 5 milimetrů za minutu. Měřící smyčka běží, dokud není přijat „STOP“ příkaz od uživatele.

4.1.2 Softwarová realizace stavového automatu

Kód ať už v jakémkoliv jazyce má vždy svoji posloupnost a je dobrým zvykem ji dodržovat a tím přispívat k čitelnosti kódu jako takového. Na začátku se nahrávají knihovny a definují konstanty a deklarují globální proměnné. Knihovny jsou kusy kódu třetích stran, které ulehčují práci. Nejznámějším příkladem je knihovna „math.h“, která obsahuje pokročilejší matematické funkce, aby je uživatel nemusel psát sám.

Na začátku jsou importovány dvě knihovny, první slouží pro ulehčení práce s externím ADC převodníkem. Knihovna „Wire.h“ je podpora pro I2C komunikaci. Tato knihovna je používaná automaticky knihovnou „Adafruit_ADS1015.h“ a proto se v kódu přímo nevyskytuje, nicméně musí být importována.

```
1. #include <TEST.h>
```

Následují definice konstant. Konstanty se v jazyce C zapisují ve tvaru „*#define název hodnota*“ a označují neměnnou hodnotu. Vyskytuje se i způsob definování konstant pomocí prefixu „const“. Rozdíl mezi direktivou a definováním konstanty pomocí prefixu je v tom, jak se proměnná poté vyskytuje v kódu. Všechny konstanty označené jako #define jsou preprocesorem nahrazeny přímo jejich hodnotou ještě před překladem kódu.

Všechny příkazy, které začínají mřížkou se nazývají direktiva preprocesoru. Dalšími direktivy jsou například: #if, #endif, # ifdef, #else… Tyto podmínky se označují jako podmíněný překlad, v praxi to znamená to, že nějaký blok kódu je přeložen pouze pokud je podmínka splněna. Výhodou je, že všechna direktiva vyhodnotí preprocesor ještě před přeložením kódu, a do zkompilovaného kódu se vůbec nenahrají části, které nebudou použity. Tímto způsobem se dá ušetřit paměť, jelikož jsme často limitováni její velikostí.

Konstanty jsou rozděleny do tří skupin. První skupina definuje výstupní piny, číselná hodnota odpovídá přímo pinu na desce s odpovídajícím číslem. PWM_duty je nastavena na 0,5 aby střída PWM byla právě 50 %. Následují stavové konstanty, ty udržují jednotlivé stavy ve smyčce. Poté jsou definovány konstanty akční, ty určují změnu stavů.

```
1. // Pin setup
2. #define MOTOR_PWM_example 0.5
3. …
4. // state machine constants
5. #define STATE_example 0
6. …
7. // action constants
8. #define ACTION_example 0
9. …
```

Kód pokračuje definováním globálních proměnných. Globální proměnná je taková proměnná, ke které mají přístup všechny funkce v aktuálním programu, není tedy třeba je předávat jako argumenty. Globální proměnné jsou definovány mimo funkce a jsou v paměti uchovávány, dokud program neskončí. Nevýhodou je, že se mohou krýt s proměnnými lokálními, tedy definovanými uvnitř funkcí a je na programátorovi, aby si ošetřil, že se v kódu nevyskytne dvakrát proměnná se stejným názvem.

```
1. // global variables
2. static unsigned int example= 0;
```

Dále je třeba nastavit PWM. Pulsně šířková modulace se nastavuje pomocí registrů, které jsou přehledně a plně popsány v datasheetu výrobce procesoru (Atmel) [7]. Pro tuto aplikaci je použit 16bitový counter register. To znamená, že maximální hodnota, do které může počítat je 2^16 což je 65536. Tím je získán větší rozsah, než by tomu bylo u 8bitového registru, kterým tento procesor také disponuje.

Registr je rozdělený do dvou částí po 8 bitech (TCCR1A a TCCR1B)

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Obrázek 23: Registr pro nastavení PWM [7]

TCCR1A/B: Kombinace bitů WGMnn udává nastavení generování PWM. Vysvětlení funkce jednotlivých nastavení je popsáno v datasheetu, konkrétně tabulka 14-4 [7]. Kombinace bitů COMnA1 a COMnA0 určuje přepínání logické hodnoty při naplnění counter registru (Tabulka 14-3 [7]) Pole bitů CS1n slouží pro nastavení děličky pulsů krystalu. V tomto případě je nastavena na hodnotu 1, tedy žádné dělení.

ICR1 udává hodnotu časovače, při které se vygeneruje nová perioda PWM. Funkce *pinMode* nastaví výstupní pin PWM.

```
1. void setupPWM() {
2.     TCCR1A = 0;
3.     TCCR1B = 0;
4.     TCNT1  = 0;
5.     TCCR1A = _BV(COM1A1)
6.            | _BV(COM1B1)
7.            | _BV(WGM11);
8.     TCCR1B = _BV(WGM13)
9.            | _BV(CS10);
10.    ICR1   = motor_clk_pwm_period;
11.    pinMode(MOTOR_CLK_PWM_PIN, OUTPUT);
12.    }
```

Dalším krokem je obsluha sériové komunikace. Sériová linka posílá zprávy jednotlivě a ty jsou pak ukládány do bufferu. Velikost jedné zprávy je právě 8 bitu, proto je nutné zprávy rozdělovat, jelikož bude po lince posílány čísla větší než 256. Pro obsluhu byla vytvořena funkce typu void, která prochází buffer sériová linky a byty ukládá do proměnné *buff*. Poté je pomocí operace bitového shiftu složeno původní číslo, které má vždy 16 bitů.

```
1. void get_message(void){
2.   for (int i=0; i<= 1; i++ ){
3.        while (Serial.available() < 1){
4.        }
5.        buff[i] = Serial.read();
6.   }
7.   motor_clk_pwm_period = (buff[0])|(buff[1]<<8);
8.   setupPWM();
9. }
```

Následuje hlavní smyčka programu. Je ukončena pouze v případě ukončení celého programu. Na začátku je podmínka, která kontroluje, jestli po sériové lince přišla nějaká zpráva. Pokud ano, je vyhodnocena akce podle kódu zprávy.

```
1.  if (Serial.available() > 0) {
2.         action = Serial.read();
3.         if (action == ACTION_RUN_DOWN_FF) {
4.             get_message();
5.         }
6.         else if (action == ACTION_RUN_UP_FF) {
7.             get_message();
8.         }
9.         else if (action == ACTION_RUN_UP){
10.                get_message();
11.            }
12.            else if (action == ACTION_RUN_DOWN) {
13.                get_message();
14.            }
15.            else if (action == ACTION_HEART_BEAT){
16.                delayCounter = 0;
17.                send_messages();
18.            }
19.            else if (ACTION_STOP == action){
20.                disablePWM();
21.                setDirUP();
22.                state = STATE_IDLE;
23.            }
```

Dále je samotný stavový automat konstruovaný pomocí *if* a *else if* struktur. Pro případ, že by se během měření odpojil datový kabel, je po lince posílám periodicky signál, který slouží jako

kontrola připojení. Nepřijde-li signál několikrát po sobě, je motor automaticky zastaven. Toto opatření slouží bezpečnosti, aby nedošlo k nechtěnému poškození části zařízení nebo vzorku.

4.2 Struktura PC aplikace

Aplikace v PC je nejvyšším článkem celého zařízení, uživatel pomocí aplikace ovládá zbytek zařízení. Software byl napsán v programovacím jazyce Python. Dále je využita řada knihoven usnadňujících práci při programování. Z požadavků vychází, že aplikace musí být schopna vykonávat, kromě komunikace s mikrokontrolerem, ještě další úlohy jako třeba ukládání měřených dat atd.

4.2.1 Python

Python je moderní programovací objektově založený jazyk. Objektem je v pythonu vše, tedy i číselné proměnné jsou objekty. Speciálním rysem Pythonu je jeho syntaxe, Těla funkcí a cyklů se nezapisují do závorek, ale jsou definovány odsazením. Tato netradiční syntaxe přispívá k čitelnosti kódu.

Obrázek 24: Logo jazyka Python [8]

Python poprvé vyšel v roce 1990 ve verzi 1.0. Od té doby vyšly ještě dvě významné verze, software ve verzi 2 a 3. Zpětná kompatibilita mezi verzemi není zachována [9].

Python sám o sobě obsahuje velké množství knihoven, které se běžně označují jako „Python Standart Library“ a jsou součástí instalačního balíku softwaru. Ostatní knihovny se instalují běžně pomocí příkazu „pip“. Pro instalaci knihovny v nejnovější verzi stačí do příkazového řádku napsat příkaz jako na obrázku 24. Název knihovny pro příkaz pip se někdy nemusí shodovat s oficiálním názvem knihovny, proto je zpravidla uveden postup, jak knihovnu nainstalovat na webových stránkách té konkrétní knihovny. Syntaxe příkazu „pip“ je na obrázku 25.

```
$ pip install SomePackage
[...]
Successfully installed SomePackage
```

Obrázek 25: instalace dodatečných knihoven [8]

Když jsou knihovny připraveny, importují se do projektu klíčovým slovem „import“.

4.2.1.1 Python a objekty

Jak bylo zmíněno v předchozím odstavci, Python je objektově orientovaný jazyk a pro efektivní práci s jazykem je třeba dobré porozumění objektům. Objektem se v programování obecně chápe instance, která má nějaké parametry a metody a je uložená v paměti jako proměnná se svým specifickým jménem. Lze tedy definovat jeden typ objektu (třídu) a poté ho požít pro více na sobě nezávislých proměnných které budou mít jiné atributy. Atributy objektu se označují prefixem „self“. Následuje příklad jednoduché třídy s názvem „Zidle“.

```
1.  class Zidle():
2.      def __init__(self, x):
3.          self.nohy = 4
4.          self.stav = x
5.      def kondice(self):
6.          if self.stav == 'stara':
7.              print('jsem rozvrzaná')
8.          elif self.stav == 'nova':
9.             print('jsem nová')
10.
11. stara_zidle = Zidle('stara')
12. nova_zidle = Zidle('nova')
```

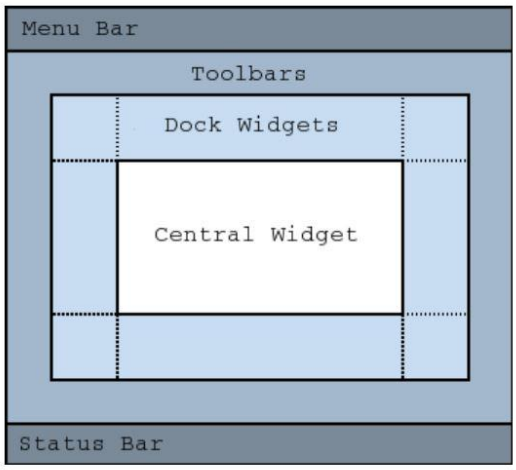
Definování objektu začíná klíčovým slovem „Class“, které definuje, že se bude jednat o třídu. Následuje funkce __init__(), jedná se o speciální funkci která inicializuje atributy objektu. Tato funkce se automaticky volá při vytváření objektu, pokud je objekt vytvářen s nějakými vstupními parametry, tak se předají právě do této funkce. V příkladu je ukázáno, že některé atributy jsou pro každý objekt stejné (počet noh), a některé atributy jsou vytvářeny se svojí speciální hodnotou.

Následuje definování funkcí, funkce uvnitř objektu se nazývají metody a mají automaticky přístup ke všem atributům objektu. Pokud jsou vytvořeny dva objekty tak jak je znázorněno v příkladu, tak při zavolání metody každého objektu se do příkazového řádku vypíše, v jaké kondici je židle.

4.2.2 PyQt5 a implementace

PyQt5 je knihovna pro tvorbu grafických uživatelských prostředí. Vychází z knihovny Qt, která je napsána pro jazyk C++. Implementace pro python je bezplatná. Jelikož je knihovna opravdu rozsáhlá, je dobré ji neimportovat celou do projektu, ale pouze moduly, které budou použity.

Pro účely této práce bylo použito byla použita třída „QMainWindow“ která slouží jako stavební kámen celé aplikace. Tento třída vytvoří prázdné grafické okénko, které se následně doplní dalšími prvky.



Obrázek 26: Rozložení okna [10]

Protože bude mít grafika aplikace vlastní metody, byla vytvořena nová třída, která přejímá metody třídy QMainWindow. Pokud dochází k dědění vlastností jedné třídy do třídy druhé, použije se funkce „super()“, která inicializuje atributy děděného objektu.

```
1. class MotorApp(QMainWindow):
2.     def __init__(self):
3.         """ Constructor """
4.         super().__init__()
```

Po této proceduře je již třída „MotorApp“ obohacena o vlastnosti třídy QMainWindow. V inicializaci probíhá vytvoření všech objektů pro grafiku a layout. Nejprve jsou vytvořena tlačítka do „Menu bar“. Tlačítko se vytvoří pomocí třídy „QAction“, pokud je třeba vytvořit vnořené tlačítko, použije se třída „QMenu“, která se následně přidá pomocí jiné metody než tlačítko.

```
1. self.File_menu_new = QAction('New Measurement', self)
2. self.File_menu_connect = QMenu('Connected to', self)
```

V ukázce je vytvořené tlačítko „New Measurement“ a menu „Connected to“. Parametr „self“ se předává metodě a slouží jako identifikátor objektu.

Když jsou nadefinovány všechny „klikátka“, je třeba jim přiřadit akce. To se dělá pomocí signálů. Signál je v Qt speciální metoda, která slouží k propojení objektů. Jestliže tedy je vytvořeno tlačítko, a uživatel na něj klikne, je pomocí signálu volána metoda objektu, která vykoná požadovanou akci.

```
1. self.File_menu_new.triggered.connect(self.set_new_measurement)
```

Teď když už má tlačítko přiřazenu svou funkci, je třeba ho přidat do menu. Samotné menu se vytvoří zavoláním „menuBar()“. Následně pomocí vnitřní metody „addMenu“ se přidá rozbalovací nabídka se jménem „File“. Do této nabídky se poté přidávají tlačítka metodou

„addAction“. Pokud je třeba do rozbalovacího menu přidat další menu, použije se metoda opět metoda „addMenu“.

```
1. self.menu = self.menuBar()
2. self.File_menu = self.menu.addMenu('File')
3. self.File_menu.addAction(self.File_menu_new)
4. self.File_menu.addMenu(self.File_menu_connect)
```

Horní menu lišta je připravena, a lze se přesunout k spodní notifikační liště. Tato lišta slouží jen k informaci, co právě program dělá. Nastaví se velmi jednoduše, pomocí metody „statusBar“ se vytvoří objekt a do něj se zapisuje stav pomocí „showMessage“. Každá funkce v sobě má řádek, který tento status aktualizuje.

```
1. self.status_bar = self.statusBar()
2. self.status_bar.showMessage('Ready')
```

Následně přichází vytvoření tlačítek na hlavní obrazovce políček pro uživatele. Postup jo obdobný jako u horní menu lišty. Nejdříve se vytvoří tlačítka, poté se přiřadí jejich funkce, a nakonec se umístí na svoji pozici.

```
1. btnRunUp = QPushButton("Run UP!")
2. btnRunUp.clicked.connect(self.btnRunUpClicked)
```

Dále je jsou vytvořena okénka se s aktuálními měřenými hodnotami. Pro tento účel byla vybrána třída „QLCDNumber“, která obsahuje digitální indikátor. Pokud se má hodnota indikátoru změnit, použije se metoda „display()“.

```
1. self.force_LCD_number = QLCDNumber()
2. self.force_LCD_number.setSegmentStyle(QLCDNumber.Flat)
3. self.force_LCD_number.setMinimumHeight(100)
4. self.force_LCD_number.setStyleSheet('border: 0 px;')
```

Posledním prvkem hlavního okna je graf, který má za účel v aktuálním čase zobrazovat měřené hodnoty. Graf je postaven na knihovně Pyqtgraph která vychází z PyQt4 a je kompatibilní s tímto grafickým prostředím. Vytvoření a nastavení grafu:

```
1. self.plotWidget = pg.PlotWidget()
2. self.plotWidget.setYRange(0, 1200)
3. self.plotWidget.setLabel('left', text='Force', units='N')
4. self.plotWidget.setLabel('bottom', text='Distance', units='m')
5. self.plotWidget.showGrid(x=True, y=True)
6. self.mForceSeries = self.plotWidget.plot()
```

Když jsou všechny prvky hlavního okna nadefinovány, je třeba je umístit. Pro toto slouží třída „QFormLayout“ která vytvoří jakousi šablonu do které se jednotlivá tlačítka a okénka umístí. Výhodou je, že pokud uživatel změní velikost okna, tak se dodrží proporcionálně rozložení okna a zachová se vizuální styl.

Vytváří se nejprve sloupce a to tak, že se vytvoří layout a přidávají se do něj řádky. Řádek se přidá voláním metody „addRow()“ a do argumentu požadované tlačítka. Takto se vytvoří layout pro každý segment, poté se tyto layouty uloží do „Widget“ objektu a uloží do „BoxLayoutu“, který rozdělí okno buď vertikálně, nebo horizontálně (záleží na jestli je volán HBoxLayout nebo VBoxLayout).

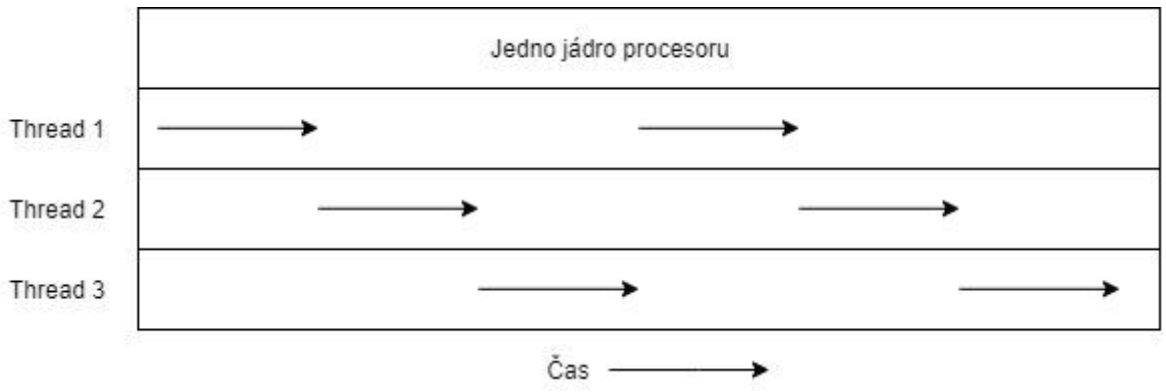
```
1. frmLayoutSetup = QFormLayout()
2. frmLayoutSetup.addRow(btnRunUp, QLabel("    "))
3. frmLayoutSetup.addRow(btnStop, QLabel("    "))
4. frmLayoutSetup.addRow(btnRunDown, QLabel("    "))
5. frmWidgetSetup = QWidget()
6. frmWidgetSetup.setLayout(frmLayoutSetup)
7.
8. layoutParameters = QVBoxLayout()
9. layoutParameters.addWidget(frmWidgetSetup)
```

Takto je celá aplikace připravena po grafické stránce k použití. Posledním krokem je aktivovat grafiku (zobrazení okna uživateli) a dát okénku název.

```
1. self.setCentralWidget(final_widget)
2. self.setWindowTitle("MotorPy")
3. self.show()
```

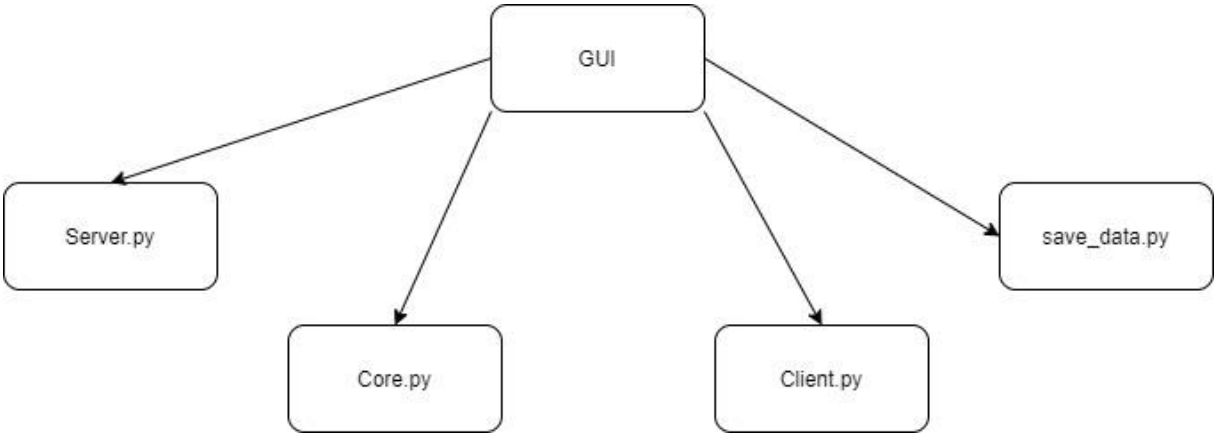
4.2.3 Struktura aplikace

Aplikace se skládá z více modulů a ty mezi sebou musí být propojené. Některé moduly zároveň musí paralelně zpracovávat svoje úlohy. Kdyby všechna funkcionalita byla na jediném vlákně, stalo by se, že než by se vykonala nějaká část v jiném než grafickém modulu, nebyl by uživatel schopen jakkoliv pracovat. Aplikace by se zdála být pomalá a nepřívětivá k používání. Aby se předešlo tomuto problému, jsou jednotlivé úlohy, které to vyžadují, provozovány na vlastním vlákně. K tomuto účelu je použita knihovna „Threading“ která tuto funkcionalitu umožňuje. Ačkoliv by se to mohlo na první pohled zdát, nejedná se o vícejádrové zpracování úlohy, nýbrž procesorový čas je rozdělován pro jednotlivé úlohy a tím se dosáhne zdání paralelního zpracovávání úlohy. Grafické znázornění tří úloh které se vykonávají na jednom jádře je na obrázku 27.



Obrázek 27: Grafické znázornění threadingu

Propojení jednotlivých modulů mezi sebou musí mít nějakou hierarchii. Vždy musí být nadřazený modul, který bude přistupovat a brát data z modulu podřazeného. Aplikace se skládá z pěti modulů. Jeden byl již popsán – grafické okénko. Dále následují moduly, které zajišťují funkcionalitu. Propojení je paralelní, grafické znázornění je na obrázku 28.



Obrázek 28: Vnitřní struktura aplikace

4.2.3.1 Jádro aplikace

Jádro aplikace slouží ke komunikaci s mikrokontrolerem. Pro jádro je vytvořena třída a je jí předán objekt umožňující pracovat s touto třídou jako se samostatným threadem.

```
1. class MotorDriveThread(threading.Thread):
2.     def __init__(self, ser):
3.         threading.Thread.__init__(self)
```

V jádru aplikace je vytvořeno pět metod, které obsluhují celý modul. První metodou je „run()“, tato metoda obstarává všechny ostatní metody a slouží jako obdoba hlavní smyčky tohoto modulu. Speciální je v tom, že právě metoda „run“ je již vytvořená v třídě „Thread“ z knihovny threading a musí mít právě tento klíčový název. Dále je v této metodě ukrytá podmínka, která kontroluje, zdali není vyžadováno ukončení tohoto vlákna. K tomu může dojít jen v případě, že je ukončena aplikace. Pokud by došlo k ukončení aplikace a neukončily se ostatní vlákna, vlákna by dále běžela na pozadí počítače a docházelo by k problémům s opětovným spuštěním aplikace.

Funkce k obsluze komunikace s mikrokontrolerem:

send_command(self) - tato funkce slouží k posílání dat přes sériovou linku do mikrokontroleru. Posílá pouze uživatelské příkazy.

send_command_heartbeat(self) – Posílá signál, který udržuje mikrokontroler v definovaném stavu (měření, rychloposuv…). Tento signál je posílán každých 50 milisekund.

get_messages(self) – Metoda, která kontroluje, zdali po sériové komunikaci dorazily nějaká data. Data po lince chodí vždy v páru (hodnota z ADC, počet vygenerovaných pulsů PWM). Pokud data přišla, uloží je do proměnných, ke kterým má přístup hlavní modul a ten je pravidelně aktualizuje a vykresluje uživateli.

4.2.3.2 Ukládání dat

Pro ukládání dat byl vytvořen modul (save_data.py), který sdružuje funkce, které ukládají do konkrétních formátů. Následující tabulka nastiňuje, jak jsou data ukládána do souborů. Sloupce Alfa_x jsou data, která přicházejí z externí aplikace Alfa. Tyto data mohou reprezentovat cokoliv, neboť si uživatel v programu zvolí, co chce posílat do této aplikace.

Tabulka 2: Pořadí a formát ukládaných dat

Pořadí	Síla	Posuv	Čas	Alfa_0	…	Alfa_7
1.	0.5	1	0	0.001	…	0.001
2.	1	2	0.01	0.001	…	0.001
. . .	. . .	. . .	. . .	. . .	… … …	. . .

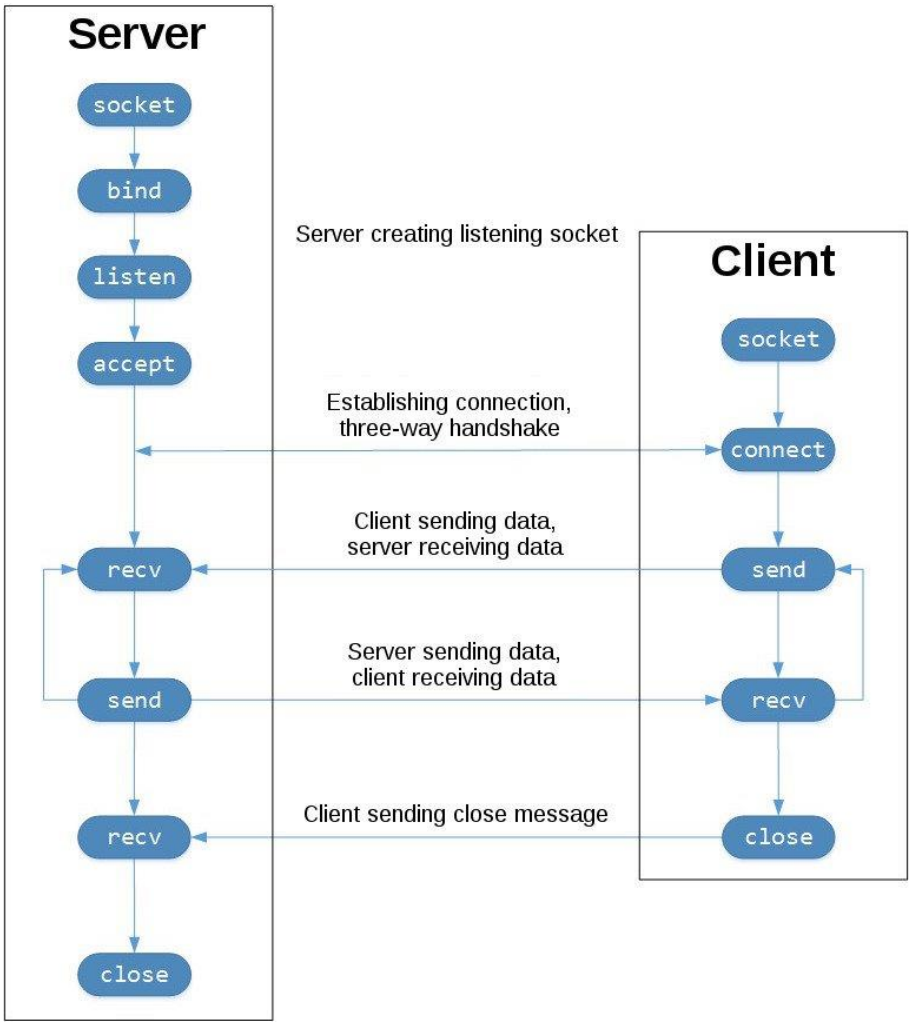
to_csv(…) – tento formát je specifický tím, že data jsou oddělena čárkou (nebo jakýmkoliv specifickým znakem). Lze do něj tedy přehledně ukládat data formou vektorů. CSV má velkou podporu ostatních hojně využívaných aplikací jako je například Microsoft Excel.

to_txt(…) – Klasický textový soubor, který otevře jakýkoliv textový prohlížeč. Kódování textu je UTF-8.Struktura ukládaných dat je opět stejná jako u CSV, tedy sloupce jsou oddělené středníkem.

to_mat(…) – Jelikož se běžně ke zpracování dat v technické praxi využívá program MATLAB, byla přidána podpora i pro tento formát. Vektory jsou ukládány do jednoho souboru. Po načtení do workspace jsou data rozbalena přímo do jednotlivých vektorů s příslušným jménem.

4.2.3.3 Komunikace s externí aplikací Alfa

Jedním z požadavků je, aby bylo možné se připojit k externí aplikaci a sdílet s ní data. Alfa má v sobě zabudovaného virtuálního klienta a server. Stejný přístup byl použit i v této aplikaci. K tvorbě síťového serveru a klienta byla použita knihovna „socket“.



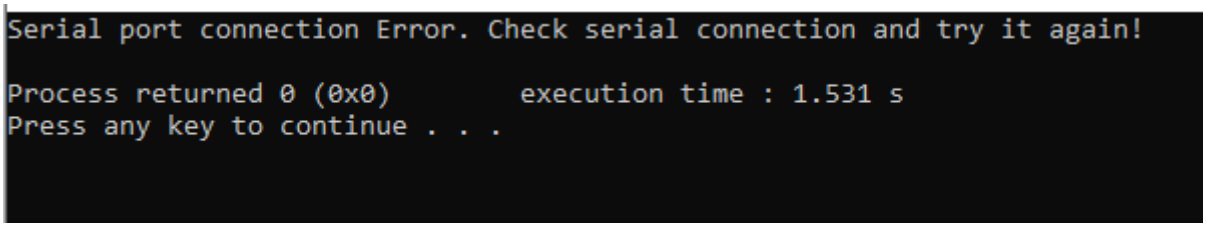
Obrázek 29: Grafické znázornění funkce klienta se serverem [11]

TCP_server.py – Modul má své vlastní vlákno a obsahuje kompletní server. Server se nachází na portu číslo 7256. Klient musí mít nastavené, že se snaží připojit na tento port, jinak se připojení nepovede. Server opakovaně kontroluje, jestli se nesnaží nějaký klient připojit a pokud ano, připojení proběhne automaticky.

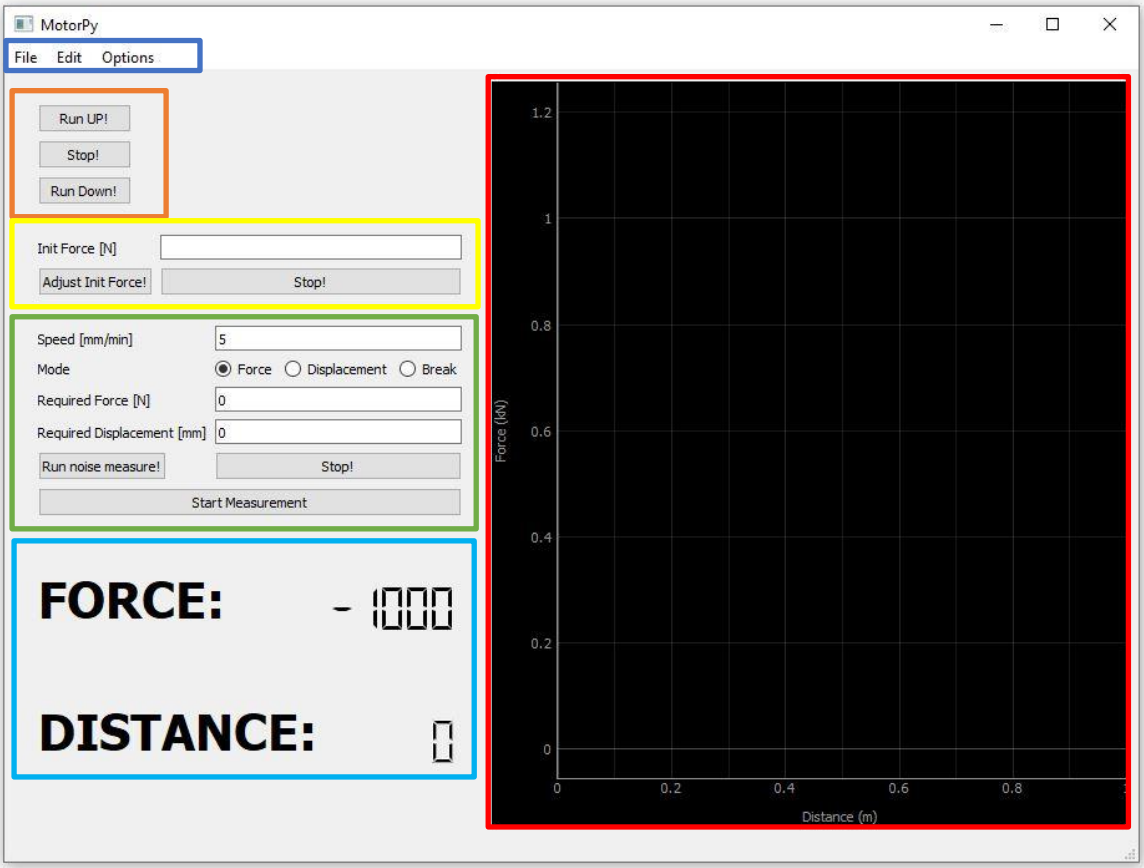
TCP_client.py – Klient stejně jako server běží na svém vlastním vlákně. Klient, podobně jako server, se snaží připojit k hostitelskému serveru na portu 700. Pokus o připojení probíhá automaticky. Pokud dojde k připojení, jsou posílána data, když se server odpojí, opět se klient snaží připojit.

4.3 Grafická stránka aplikace

V této kapitole bude popsána aplikace z pohledu uživatele tak, aby byl schopen s ní pracovat. Před spuštěním aplikace musí být připojen komunikační kabel mezi počítačem a zařízením. Pokud neexistuje spojení, aplikace se nespustí a vyskočí chybová hláška:



Obrázek 30: Chyba komunikace s hardwarem



Obrázek 31: Hlavní uživatelské okno

Hlavní uživatelské okénko je na obrázku 31. a je rozděleno do několika sekcí podle barev. Každá sekce bude popsána zvlášť.

4.3.1.1 Oranžový výběr

Tato sekce je pro ovládání stroje v režimu rychloposuvu. Obsahuje tři tlačítka:

Run Up – rychloposuv směrem nahoru

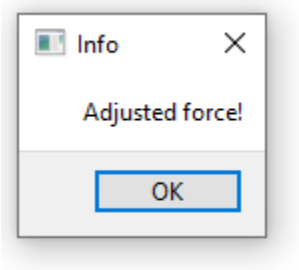
Stop – zastavení stroje

Run Down – rychloposuv směrem dolů

Pokud je zařízení v režimu rychloposuvu, nedochází k ukládání dat.

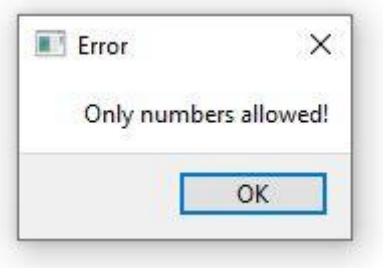
4.3.1.2 Žlutý výběr

Žlutě ohraničená sekce slouží k nastavení předpětí. První řádek označený jako „Init Force“ slouží pro uživatelem zadanou hodnotu předpínací síly. Druhý řádek obsahuje tlačítko „Adjust Init Force“, které spustí měření. Když síla na čelistech dosáhne požadované hodnoty, stroj se automaticky zastaví a objeví se informativní okénko:



Obrázek 32: Informace o dosažení předpínací síly

Pokud uživatel zadá jiné než číselné znaky a pokusí se spustit měření, objeví se chybová hláška:



Obrázek 33: Chybová hláška, zadána nečíselná hodnota

Po odkliknutí chybové hlášky se z okénka vymažou všechny znaky.

4.3.1.3 Zelený výběr

Zeleně ohraničená část slouží pro samotné měření. Základní rychlost posuvu je 5 mm/min. Tato rychlost je nastavena vždy po spuštění programu nehledě na to, co uživatel nastavil při předchozím používání – všechna nastavení se automaticky obnovují do základního režimu po spuštění aplikace. Uživatel si v řádku „Speed“ může nastavit svoji vlastní měřící rychlost. Opět platí, že pokud je zadána nečíselná hodnota, vyskočí chybová hláška. Chybová hláška má v tomto případě stejnou podobu jako na obrázku 27.

Další řádek obsahuje výběr, zdali se zařízení má zastavit na konkrétní tahové síle, nebo až na konkrétní vzdálenosti. Pokud je vybrána volba „Force“, je třeba před spuštěním zadat do kolonky „Required Force“ sílu. Síla je zadávána v základních jednotkách, tedy newtonech. Pokud je vybrána volba „Displacement“, je před měřením třeba zadat hodnotu do políčka „Required Displacement“. Vzdálenost se zadává v milimetrech z důvodu přehlednosti (pouze malé posuvy). Pokud je zadán nečíselný znak, objeví se chybová hláška jako na obrázku 27. Vzdálenost je měřena vždy relativně od místa, kde bylo spuštěno měření.

4.3.1.4 Světle modrý výběr

Modře ohraničená oblast je indikace měřené síly a uražené vzdálenosti. Síla je měřena od počátku spuštění aplikace a slouží pro informaci uživateli, aby věděl, jestli nemá nějaké zatížení už před samotným měřením. Vzdálenost je pouze pro informaci a je zobrazována jen když je aktivní měření.

4.3.1.5 Fialový výběr – graf

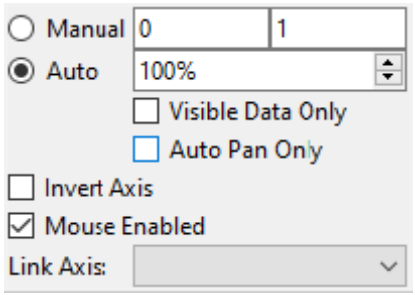
Graf zabírá celou pravou polovinu okna, právě proto, aby měřená data byla dostatečně viditelná. Graf je interaktivní a lze se v něm pohybovat pomocí kliknutím a podržením levého tlačítka myši. Jednotky grafu se mění s tím, jak se mění přiblížení. Přiblížení a oddálení může uživatel provést kolečkem myši. Po spuštění aplikace je vždy na svislé ose grafu síla a na vodorovné ose vzdálenost. Tyto osy se dají změnit v nastavení.

Kliknutím pravého tlačítka myši se rozbalí následující nabídka:

Obrázek 34: Manu nabídka grafu

1) Kliknutím na tuto volbu se graf vycentruje tak, aby byl vykreslený všechen jeho obsah.

2 a 3) Po rozbalení této nabídky se zobrazí okénko (obrázek č.35). Okénko obsahuje nastavení pro konkrétní osu. První volba je Auto nebo Manual, jedná se o nastavení rozsahu pro osu. Když je zadáno Manual, osa bude vykreslena dle zadaného rozsahu. Pokud je nastaveno Auto, osa se přizpůsobí podle rozsahu (rozsah se zadává v procentech). Procento určuje, kolik dat ze všech vykreslených bude zobrazeno.



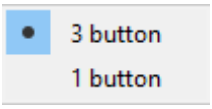
Obrázek 35: Nastavení osy v grafu

Pokud je zatrhnuto Visible Data Only, tak automaticky bude graf centrován tak, aby byla vidět data na příslušné ose. Pokud je zaškrtnuto Auto Pan Only, bude se graf centrovat do středu grafu, ale nebude se měřit jeho rozlišení.

Invert Axis – otočí všechna data na příslušné ose (záporná budou na pravé straně).

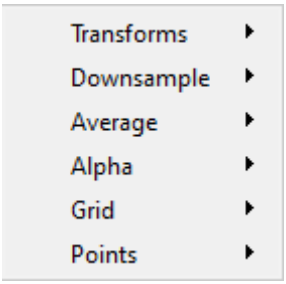
Mouse Enabled – když je zaškrtnuté, lze se podél této osy posouvat myší.

4) Mouse mode, mění režim používání myši, po rozbalení nabídky má uživatel na výběr ze dvou možností. V základu je vždy nastavena první volba, stejně jako je tomu na obrázku 36. V této volbě funguje levé tlačítko myši jako posouvač v prostoru, kolečko přibližuje a oddaluje graf a pravé tlačítko při podržení a táhnutí mění rozlišení jednotlivých os v závislosti na směru, kterým uživatel táhne.



Obrázek 36: Nastavení myši

5) Nastavení grafu, obsahuje vlastní nabídku (obrázek 37).

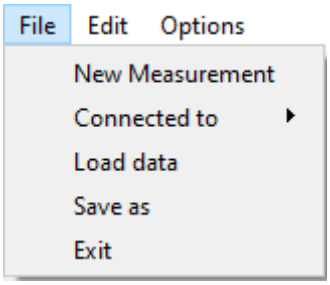


Obrázek 37: Nastavení grafu

4.3.1.6 Tmavě modrý výběr – horní lišta

Horní lišta obsahuje zbytek nastavení a dalších možností, které nejsou na hlavní obrazovce. Obsahuje tři položky a každá z položek má svoje volby.

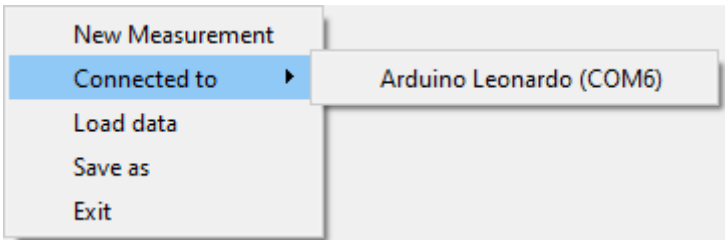
File – tento výběr obsahuje informace a volby pro měření.



Obrázek 38: File menu

New Measurement – po stisknutí tohoto tlačítka se spustí nové měření. Po spuštění programu jsou všechny vnitřní proměnné pro měřená data prázdné, pokud uživatel měří data a pak chce měřit nový soubor dat, musí nejprve stisknout toto tlačítko. Pokud tak neučiní, data se budou neustále ukládat do jednoho vektoru.

Connected to – je informativní menu, které zobrazuje připojené zařízení. Pokud je připojen kontrolér, je zobrazeno jeho jméno.



Obrázek 39: Informace o připojeném zařízení

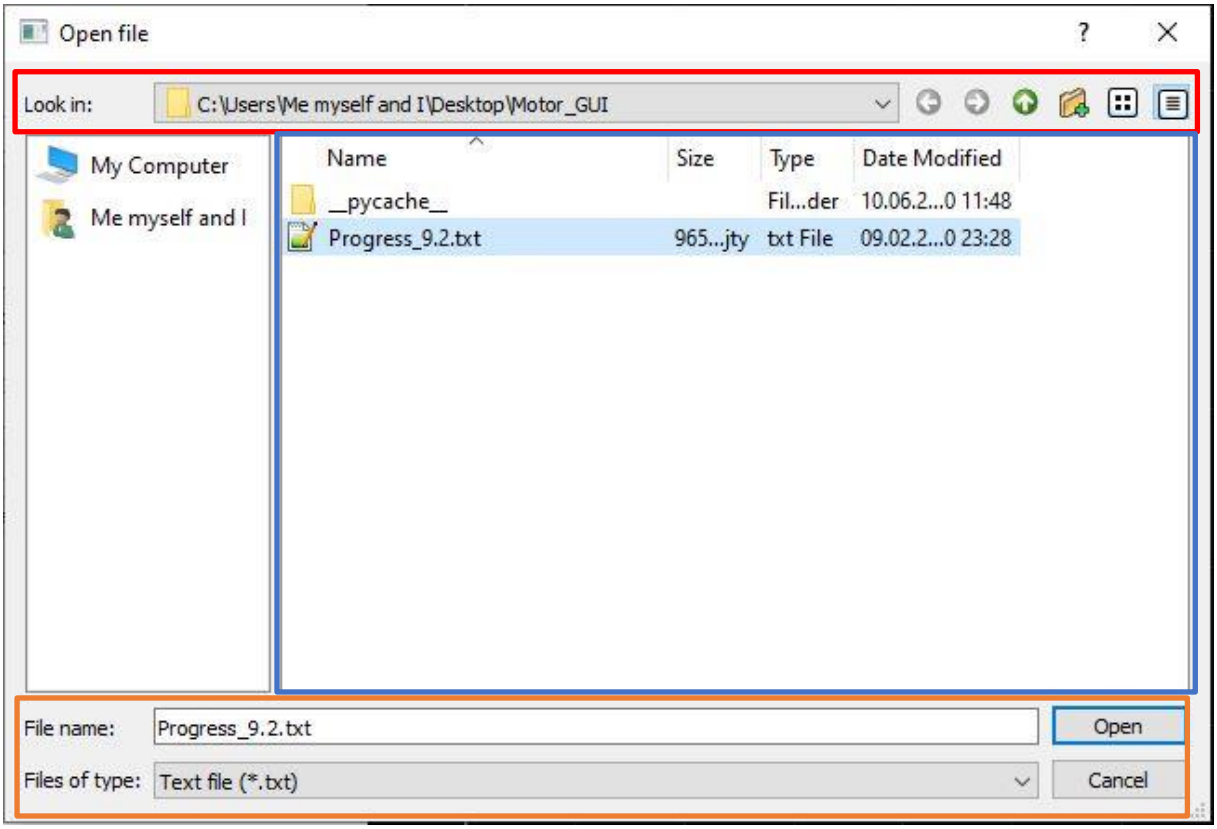
Load data – Slouží pro načítání dat do programu. Data jsou načítána z textového souboru, nebo souboru s oddělovači (CSV). Oddělovače musí být středníky. Načítaná data jsou podporována ve stejném tvaru jako jsou ukládána, viz předchozí kapitola, podnadpis „ukládání dat“.

Po stisknutí tlačítka se objeví dialogové okno (obrázek 40.).

Červené ohraničení – Navigace v adresáři, řádek je rozbalovací. Lze v něm intuitivně hledat, nebo do něj přímo napsat či vložit cestu ke konkrétnímu adresáři. Na konci řádku se nachází několik ikon, které taktéž slouží pro pochybování v adresářích, vytváření nových složek a změnu zobrazení položek v aktuálním adresáři.

Modré ohraničení – Obsah aktuálního adresáře. Obsah je filtrovaný, zobrazují se složky a soubory dle výběru v oranžovém okně.

Oranžové ohraničení – první řádek obsahuje políčko pro text – název souboru i s příponou. Druhý řádek obsahuje rozbalovací menu s typy souborů, které lze otevřít. Pokud je v tomto políčku vybrán nějaký typ souboru, je filtrován i náhled a nezobrazují se jiné typy.

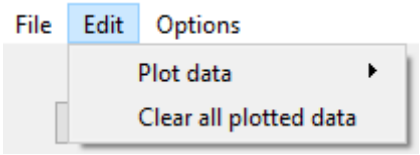


Obrázek 40: dialogové načítací okno

Save data – Tlačítko pro uložení dat. Po stisknutí se otevře stejné okno jako je na obrázku 39. Uživatel má na výběr z uložení do tří formátů (TXT, CSV, MAT). Data, co mají být uložena jsou vždy ta data, která jsou měřená.

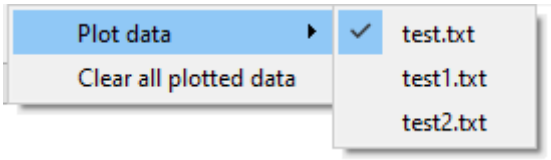
Exit – Ukončí aplikaci, má stejnou funkci jako červený křížek.

Edit – obsahuje možnosti pro vykreslování dat v grafu.



Obrázek 41: Edit menu

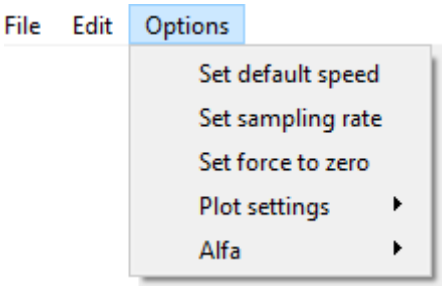
Plot data – Pokud jsou v aplikaci načtena nějaká data, zobrazí se v této nabídce. Jméno dat je stejné jako je jméno souboru, ve kterém jsou uložena. Zatrhnutím dojde k vykreslení dat do grafu, je možné vykreslovat více dat do jednoho grafu.



Obrázek 42: Zobrazení načtených dat

Clear all plotted data – Odstraní z grafu všechna vykreslená data, data však zůstanou nahraná v paměti.

Options – Toto menu obsahuje nastavení pro mikrokontroler a také dodatečné nastavení pro graf.



Obrázek 43: Option menu

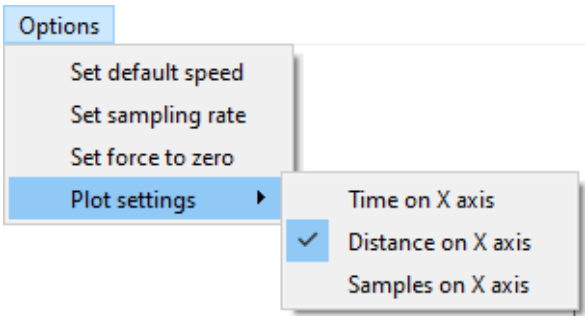
Set default speed – Tato volba slouží pro změnu rychlosti rychloposuvu zařízení. Po kliknutí se zobrazí následující okénko (obrázek 44), do kterého uživatel zadá požadovanou rychlost. Rychlost je omezená na maximální hodnotu 100 mm/minutu. Tato rychlost je více než dostačující pro orientační přiblížení čelistí.

Obrázek 44: Dialogové okno pro zadávání rychlosti posuvu

Set sampling rate – Tato volba slouží pro změnu rychlosti logování. Snížením rychlosti se bude zaznamenávat méně vzorků a data budou zabírat méně místa na disku. Po stisknutí tlačítka se objeví dialogové okno podobné jako tomu na obrázku 44.

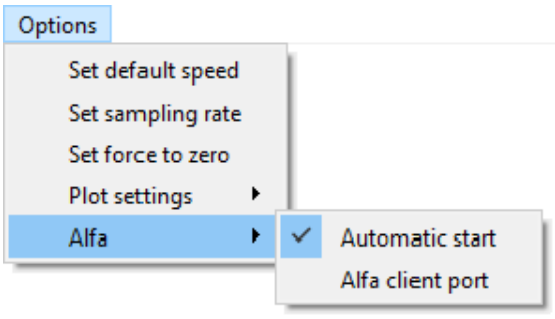
Set force to zero – Tlačítko sloužící k vynulování měřené síly. Pokud je i při nezatíženém stavu měřená nějaká nenulová hodnota, toto tlačítko nastaví nulu.

Plot settings – Nastavení vodorovné osy grafu. Po spuštění programu je vždy nastavena výchylka. Dále se dá zvolit čas, anebo počet vzorků.



Obrázek 45: Nastavení osy grafu

Alfa – Nastavení pro externí program Alfa. Tento program slouží pro digitální obrazovou korelaci. Po kliknutí na tuto možnost se rozbalí menu. V menu je políčko „Automatic start“, toto políčko říká, zdali je poslán signál pro spuštění Alfy při spuštění měření. Defaultně je políčko zaškrtnuté.

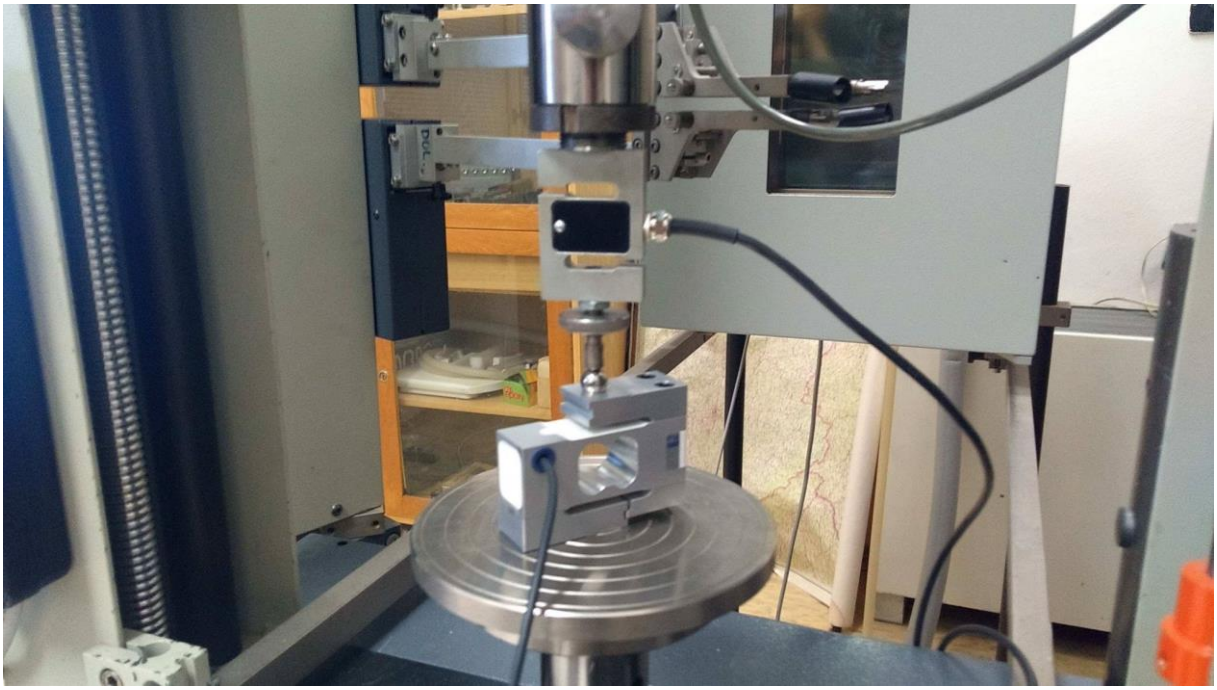


Obrázek 46: Nastavení Alfa

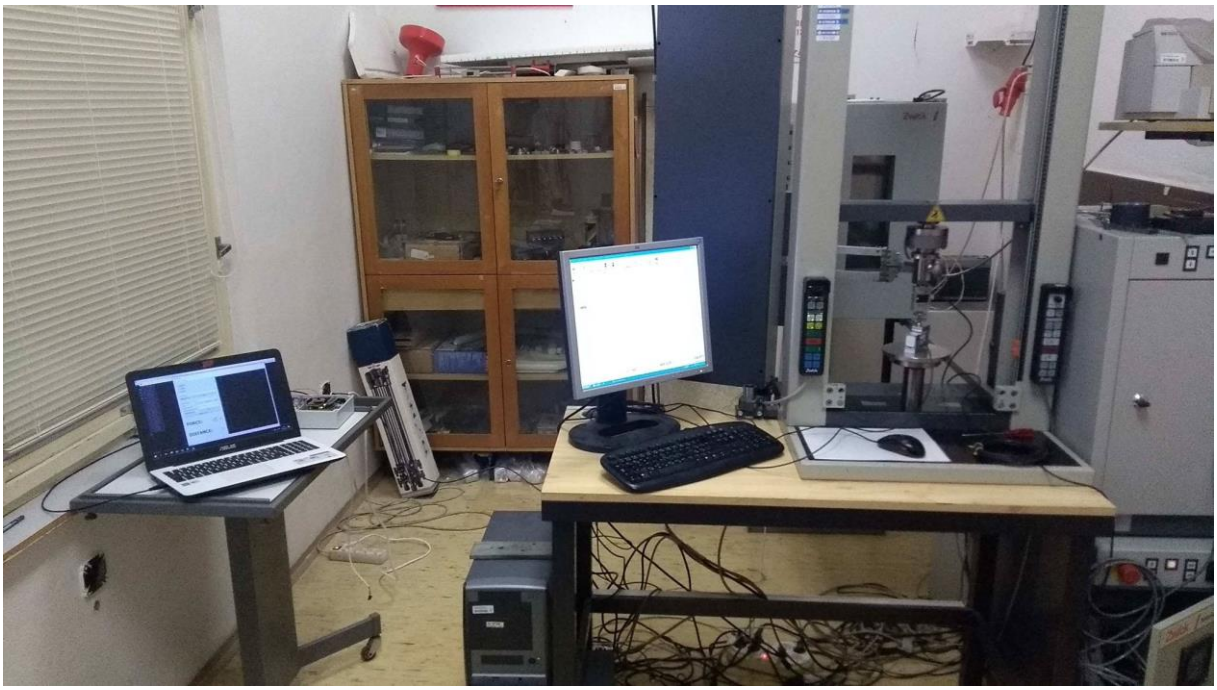
Volba „Alfa client port“ otevře dialogové okno, které nechá uživatele nastavit port pro hledání serveru. Defaultní port je 400.

5 Ověření funkčnosti

Zařízení bylo kalibrováno na přesné tenzometrické hlavě. Nejprve byl tenzometr vyjmut ze stroje, a v nezatíženém stavu se odečetl offset který způsobují nepřesnosti v elektronice. Poté pro byl tenzometr zatížen na hodnotu 10 N. Ze vstupního napětí a zatížení bylo určeno počet voltů na newton. Přepočet v softwaru byl upraven a následným zatížením bylo ověřeno, že měří správně.



Obrázek 47: Kalibrace softwaru za pomoci přesné tenzometrické hlavy

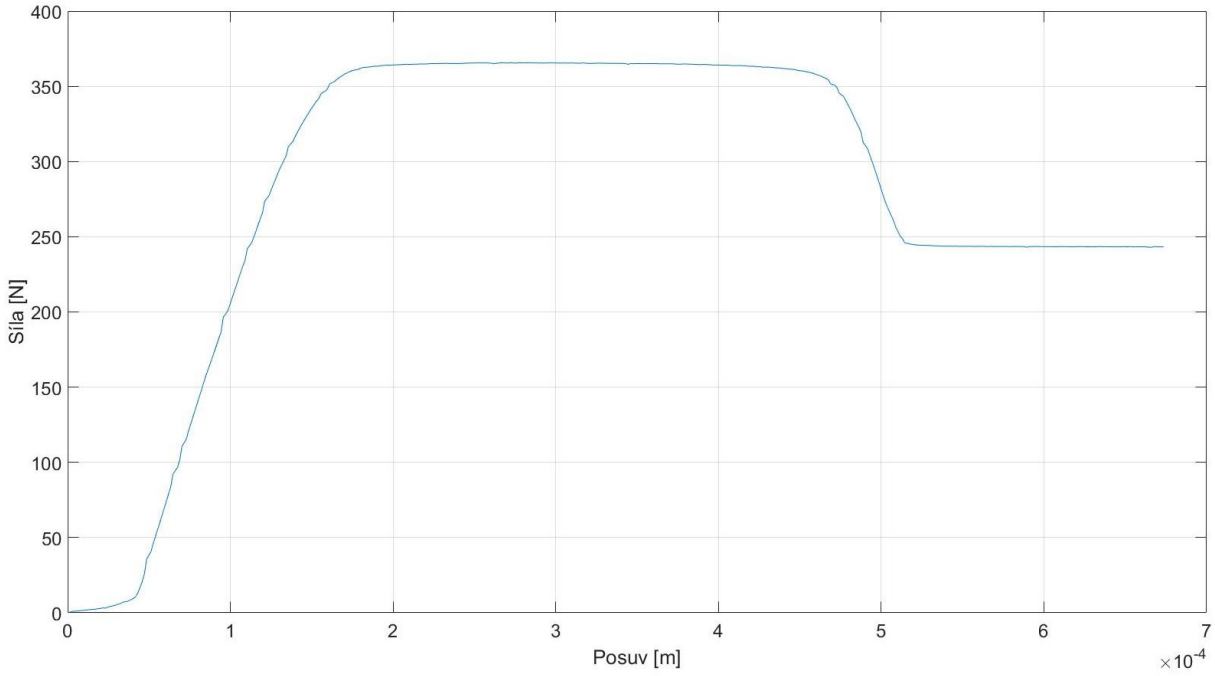


Obrázek 48: Kalibrování softwaru v laboratoři

K testu byly připraveny vzorky z plechu o tloušťce 0,2 mm z neznámého materiálu. Vzorky byly připraveny ručně, tudíž není známa síla, při které dojde k roztržení. Jelikož šlo o ruční výrobu, je každý vzorek originál a má trochu jiné rozměry.

Obrázek 49: Vzorek upnutý v čelistech

Vzorek byl upnutý do čelistí a bylo spuštěno měření. Naměřen byl následující průběh zatížení:



Obrázek 50: Naměřený průběh zatížení na vzorku

V první části grafu zatížení lehce stoupá, to je způsobené tím, že vzorek po upnutí do čelistí není předepjatý, dochází tedy pouze k vymezení všech vůlí v zařízení. Poté co začne zatížení prudce stoupat, už dochází k napínání vzorku. Jak se začíná vzorek trhat, zatížení se již nezvětšuje a je přibližně konstantní. Po utržení vzorku zůstane měřená síla nenulová, to je způsobené chybou konstrukce. Předepnutí se odstraní lehkým zatížením spodních čelistí opačným směrem.

6 Závěr

V této diplomové práci je popsán návrh řídicího systému pro malý zkušební stroj. Na toto zařízení byly kladeny požadavky z hlediska funkcionality a uživatelské přívětivosti.

Pro realizaci byla dodána konstrukce, pohonná jednotka a tenzometr. Po seznámení se s dodanými komponentami byl vybrán zbytek hardwaru. Jako mikrokontroler bylo zvoleno Arduino Leonardo, které je vhodné pro svoji jednoduchost a širokou podporu knihoven. Dále byla použita tenzometrická měřicí stanice od firmy HBM. Komunikace s řídicí jednotkou je zajištěna pomocí vstupně výstupních pinů. Komunikace mikrokontroleru s počítačem je po USB.

Softwarové řešení se děli na dvě části. První částí je software mikrokontroleru, ten byl napsán v jazyce C. Jako jednoduchým a vhodným řešením se ukázalo použití stavového automatu. Tento stavový automat neustále obsluhuje všechny činnosti hardwaru a komunikaci. Důraz byl kladen i na bezpečnost, a proto je zakomponovaná softwarová kontrola připojení. Druhou částí je uživatelské rozhraní. Toto rozhraní bylo napsáno v jazyce Python. Obsahuje veškeré uživatelské funkce k obsluze zařízení. Umožňuje načítat a ukládat data, měnit rychlost pojezdu atd.

Posledním bodem práce je ověření funkčnosti stroje. Nejprve bylo nutné udělat kalibraci měřené síly. Pro toto byl tenzometr zatížen přesně definovanou silou na zkušebním stroji. Poté byl přepočet v softwaru upraven. Následně se ověřila funkčnost celé sestavy. Pro tento účel byly vytvořeny zkušební vzorky. Vzorky byly ručně vyrobené a sloužily čistě pro demonstraci funkčnosti. Materiál vzorku není známý. Z experimentu byl naměřen tahový diagram, který je na obrázku 50.

7 Bibliografie

[1] SANCHEZ, Julio a Maria P CANTON. *Embedded Systems Circuits and Programming*. 1. CRC Press, 2012. DOI: 10.1201/b11899. ISBN 9781439879047.

[2] *Schneider Electric Motion USA: Datasheet*. 8. 2011.

[3] *S2M Force transducer: Datasheet* [online]. B3594-1.3. Hottinger Baldwin Messtechnik [cit. 2020-06-04]. Dostupné z: www.hbm.com

[4] *Industrial Amplifier: Datasheet* [online]. A0114-6.3. Hottinger Baldwin Messtechnik [cit. 2020-06-04]. Dostupné z: www.hbm.com

[5] In: *Arduino.cc* [online]. 2020 [cit. 2020-06-04].

[6] *LRS-100: Datasheet* [online]. 1. Mean Well, 2015 [cit. 2020-06-04]. Dostupné z: www.meanwell.com

[7] *ATmega16U4/32U4: Datasheet* [online]. Rev. 7766J. San Jose: Atmel, 2016 [cit. 2020-06-04]. Dostupné z: https://www.microchip.com/wwwproducts/en/ATmega32u4

[8] *Python* [online]. Python Software Foundation [cit. 2020-06-05]. Dostupné z: Python.org

[9] *Python* [online]. [cit. 2020-06-04]. Dostupné z: http://en.wikipedia.org/wiki/Python

[10] *PyQt5* [online]. Riverbank Computing Limited [cit. 2020-06-05]. Dostupné z: https://pypi.org/project/PyQt5/

[11] *Socket Programing in Python* [online]. [cit. 2020-06-07]. Dostupné z: https://realpython.com/python-sockets/

8 Seznam obrázků

9 Seznam tabulek